The eoapplication Package

API Reference

DRAFT

Apple Computer, Inc.

© 2000 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Computer, Inc., except to make a backup copy of any documentation provided on CD-ROM.

The Apple logo is a trademark of Apple Computer, Inc.
Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this book. Apple retains all intellectual property rights associated with the technology described in this book. This book is intended to assist application developers to develop applications only for Apple-labeled or Apple-licensed computers.

Every effort has been made to ensure that the information in this manual is accurate. Apple is not responsible for typographical errors.

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Macintosh, and WebObjects are trademarks of Apple Computer, Inc., registered in the United States and other countries.

Enterprise Objects is a trademark of Apple Computer, Inc.

NeXT, the NeXT logo, OPENSTEP, Enterprise Objects Framework, Objective–C, and WEBSCRIPT are trademarks of NeXT Software, Inc.

Adobe, Acrobat, and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions.

Helvetica and Palatino are registered trademarks of Linotype-Hell AG and / or its subsidiaries.

ITC Zapf Dingbats is a registered trademark of International Typeface Corporation.

ORACLE is a registered trademark of Oracle Corporation, Inc.

SYBASE is a registered trademark of Sybase, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X / Open Company Limited.

Windows NT is a trademark of Microsoft Corporation.

All other trademarks mentioned belong to their respective owners.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

The eoapplication Package

Package: com.apple.client.eoapplication

Introduction

Documentation for this package is forthcoming. For information on using this package, see the book *Getting Started with Direct to Java Client*.

The most important classes in this package are:

Class	Description
EOController	A representation of controller objects responsible for managing and sometimes generating the user interface of a Java Client application
EOComponentController	A controller that manages user interface components
EOEntityController	A controller that displays enterprise objects in a user interface
EODocumentController	A controller that displays and edits enterprise objects in a user interface
EOInterfaceController	A controller that represents an Interface Builder nib file
EOApplication	The Java Client application
EOAppletController	A representation of an EOApplet as a controller in the controller hierarchy

Draft.. © Apple Computer, Inc.

Class	Description
EOApplet	The default Applet class used in Java Client applications
EOAction	An abstract representation of operations the user can invoke from the user interface
EOXMLUnarchiver	An object containing the parameters from an XML specification used to create the controllers in the controller hierarchy

Rule System and XML Description

For Direct to Java Client applications, the controller hierarchy for the client side of the application is created from an XML description. The XML description is created by a rule system on the server side of the application. The details of this process are described in the book *Getting Started With Direct to Java Client*.

There are three pieces of information associated with an EOController class that are used to generate a controller hierarchy with the rule system: the controller's `controllerType`, its corresponding XML tag, and its corresponding XML attributes. Each controller class specification identifies this information in a section titled "Rule System and XML Description".

controllerType

You use the `controllerType` key to define custom rules that should fire only for certain kinds of controllers. For example, suppose you want to set the minimum width of all an application's windows. To do so, you write a rule whose condition specifies that the `controllerType` is 'windowController'. Then the rule only fires for controllers that control windows. Each controller falls into one of the following controller types:

- windowController
- modalDialogController
- entityController
- widgetController
- tableController
- groupingController
- dividingController

Draft.. © Apple Computer, Inc.

- actionWidgetController

XML Tag and Attributes

As an XML description is parsed, an EOXMLUnarchiver maps XML tags to particular EOController classes. All concrete controller classes—classes whose instances can actually be used in a controller hierarchy—have an XML tag.

XML attributes tell the EOXMLUnarchiver how to configure the controllers. XML attributes are inherited. For example, there are three XML attributes defined for EOController—className, disabledActionNames, and typeName. These attributes can be used by any controller, however, because all controllers are subclasses of EOController.

Draft.. © Apple Computer, Inc.

Draft.. © Apple Computer, Inc.

BeansAppletContext

Inherits from: Object

Implements: java.applet.AppletContext

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

`getApplet`

`getApplets`

`getAudioClip`

`getImage`

`showDocument`

Draft. © Apple Computer, Inc.

showDocument

showStatus

Instance Methods

getApplet

```
public java.applet.Applet getApplet(String aString)
```

getApplets

```
public java.util.Enumeration getApplets()
```

getAudioClip

```
public java.applet.AudioClip getAudioClip(java.net.URL anURL)
```

getImage

```
public synchronized java.awt.Image getImage(java.net.URL anURL)
```

showDocument

```
public void showDocument(
   java.net.URL anURL,
   String aString)
```


Draft. © Apple Computer, Inc.

showDocument

```
public void showDocument(java.net.URL anURL)
```

showStatus

```
public void showStatus(String aString)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

BeansAppletStub

Inherits from: Object

Implements: java.applet.AppletStub

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

`appletResize`

`getAppletContext`

`getCodeBase`

`getDocumentBase`

`getParameter`

Draft. © Apple Computer, Inc.

isActive

Instance Methods

appletResize

```
public void appletResize(
   int anInt,
   int anInt)
```

getAppletContext

```
public java.applet.AppletContext getAppletContext()
```

getCodeBase

```
public java.net.URL getCodeBase()
```

getDocumentBase

```
public java.net.URL getDocumentBase()
```

getParameter

```
public String getParameter(String aString)
```


Draft. © Apple Computer, Inc.

isActive

```
public boolean isActive()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

BeansCallback

Inherits from: Object

Implements: _EOTimer.Callback

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Instance Methods

timerFired

```
public void timerFired(
   _EOTimer a_EOTimer,
   java.util.Date aDate,
   java.util.Date aDate)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOAction

Inherits from: javax.swing.AbstractAction

Package: com.apple.client.eoapplication

Class Description

EOAction objects are abstract representations of operations the user can invoke from the user interface. An action does not specify how it appears in the user interface—it can appear as a button, a menu item, or both.

Each action defines a method called the action name, that is invoked when the action triggers. An action also has a description path, which describes the category of the action and its name. For example, a Quit action's description path might be "`Document/Quit`". In addition, the action can have a short description that differs from the last element of the description path, for example, "`Quit the Application`".

Actions can have icons for buttons in the application and small icons for minor buttons in the user interface. To allow users to trigger actions with "hot-keys," each action has a menu accelerator, a javax.swing.KeyStroke the user can type on the keyboard.

Actions often appear in groups in the user interface: buttons in the same group are rendered close together and menu items in the group are rendered in separate menus like the Document, Edit, Tools, or Window menus. To group actions, EOAction defines a category priority. All actions in the same group have the same category priority. An additional parameter, the action priority defines the order in which actions appear within a group (for example, the order menu items appear within a menu).

Draft. © Apple Computer, Inc.

An action triggers when the user clicks the corresponding user interface widget. In most cases, the action's method is dispatched to the subcontrollers of the controller that displays the action. Methods whose names end with `...ForControllerHierarchy` return such actions. In some cases, the action's method is dispatched to the active widget, like the text field containing the cursor. Methods whose names end with `...ForFocusComponent` return such actions. In other cases, the action's method is dispatched to a particular object, usually the EOApplication at the root of the controller hierarchy.

EOAction defines methods to create actions, access an action's parameters, manage groups of actions, and accessing shared actions used in Direct to Java Client applications.

Method Types

Accessing action parameters

- `actionName`
- `actionPriority`
- `actionTitle`
- `categoryPriority`
- `descriptionPath`
- `descriptionPathComponents`
- `icon`
- `menuAccelerator`
- `setActionName`
- `setActionPriority`
- `setCategoryPriority`
- `setDescriptionPath`
- `setIcon`
- `setMenuAccelerator`
- `setShortDescription`

Draft. © Apple Computer, Inc.

setSmallIcon

shortDescription

smallIcon

Creating actions

EOAction

actionForControllerHierarchy

actionForFocusComponent

actionForObject

standardActionForFocusComponent

standardDocumentActionForControllerHierarchy

standardDocumentActionForApplication

standardDocumentActionForControllerHierarchy

standardEditActionForControllerHierarchy

Creating menu accelerators

keyStrokeWithKeyCode

keyStrokeWithKeyCodeAndModifiers

keyStrokeWithKeyCodeAndShiftModifier

keyStrokeWithString

Accessing specific shared actions

standardActivatePreviousWindowActionForApplication

standardAddActionForControllerHierarchy

standardAppendActionForControllerHierarchy

standardCancelActionForControllerHierarchy

standardClearActionForControllerHierarchy

standardCloseWindowActionForControllerHierarchy

standardDeleteActionForControllerHierarchy

standardDeselectActionForControllerHierarchy

Draft. © Apple Computer, Inc.

standardEditActionsForFocusComponent

standardFindActionForControllerHierarchy

standardInsertActionForControllerHierarchy

standardInsertWithTaskActionForControllerHierarchy

standardOkActionForControllerHierarchy

standardOkAndSaveActionForControllerHierarchy

standardOpenWithTaskActionForControllerHierarchy

standardQuitActionForApplication

standardRedoActionForControllerHierarchy

standardRefreshActionForApplication

standardRemoveActionForControllerHierarchy

standardRevertActionForControllerHierarchy

standardSaveActionForControllerHierarchy

standardSaveAllActionForApplication

standardSelectActionForControllerHierarchy

standardUndoActionForControllerHierarchy

Managing actions

actionCanBePerformedInContextOfController

actionPerformed

mergedActions

sortedActions

Managing the property change listener

addPropertyChangeListener

firePropertyChange

removePropertyChangeListener

Methods inherited from Object

equals

Draft. © Apple Computer, Inc.

toString

Constructors

EOAction

```
public EOAction(
   String actionName,
   String descriptionPath,
   String shortDescription,
   javax.swing.Icon icon,
   javax.swing.Icon smallIcon,
   javax.swing.KeyStroke menuAccelerator,
   int categoryPriority,
   int actionPriority)
```

Returns a new action (an EOAction object) as specified by the arguments.

See Also: actionName, descriptionPath, shortDescription, icon, smallIcon, menuAccelerator, categoryPriority, and actionPriority.

Static Methods

actionForControllerHierarchy

```
public static EOAction actionForControllerHierarchy(
   String actionName,
   String descriptionPath,
   String shortDescription,
   javax.swing.Icon icon,
   javax.swing.Icon smallIcon,
   javax.swing.KeyStroke menuAccelerator,
```


Draft. © Apple Computer, Inc.

```
   int categoryPriority,
   int actionPriority,
   boolean sendsActionToAllControllers)
```

Returns a new action (an EOAction object) as specified by the arguments. When this action triggers, it is dispatched to the subcontrollers of the controller that displays it. If *sendsActionToAllControllers* is `true`, the action is dispatched to the subcontrollers of the controller that displays the action. Otherwise, the action is dispatched to the first subcontroller that responds to it.

See Also: `EOAction`, `actionName`, `descriptionPath`, `shortDescription`, `icon`, `smallIcon`, `menuAccelerator`, `categoryPriority`, and `actionPriority`.

actionForFocusComponent

```
public static EOAction actionForFocusComponent(
   String actionName,
   String descriptionPath,
   String shortDescription,
   javax.swing.Icon icon,
   javax.swing.Icon smallIcon,
   javax.swing.KeyStroke menuAccelerator,
   int categoryPriority,
   int actionPriority)
```

Returns a new action (an EOAction object) as specified by the arguments. When this action triggers, it is dispatched to the active widget (for example, the text field containing the cursor). The other parameters are identical to the EOAction constructor parameters.

See Also: `EOAction`, `actionName`, `descriptionPath`, `shortDescription`, `icon`, `smallIcon`, `menuAccelerator`, `categoryPriority`, and `actionPriority`.

actionForObject

```
public static EOAction actionForObject(
   String actionName,
   String descriptionPath,
   String shortDescription,
   javax.swing.Icon icon,
   javax.swing.Icon smallIcon,
   javax.swing.KeyStroke menuAccelerator,
```


Draft. © Apple Computer, Inc.

```
    int categoryPriority,
    int actionPriority,
    Object object)
```

Returns a new action (an EOAction object) as specified by the arguments. When this action triggers, it is dispatched directly to *object*. To create an action that gets dispatched to the application, set *object* to the EOApplication at the top of the controller hierarchy. The other parameters are identical to the EOAction constructor parameters.

See Also: EOAction, actionName, descriptionPath, shortDescription, icon, smallIcon, menuAccelerator, categoryPriority, and actionPriority.

keyStrokeWithKeyCode

```
public static javax.swing.KeyStroke keyStrokeWithKeyCode(int keyCode)
```

Returns a KeyStroke given the numerical key code *keyCode* with the appropriate modifier for the client operating system (usually CTRL_MASK). See Sun's javax.swing.KeyStroke documentation for more information.

keyStrokeWithKeyCodeAndModifiers

```
public static javax.swing.KeyStroke keyStrokeWithKeyCodeAndModifiers(
    int keyCode,
    int modifiers)
```

Returns a KeyStroke given the numerical key code *keyCode* and the modifier mask *modifiers*. This method adds the appropriate modifier for the client operating system (usually CTRL_MASK). See Sun's javax.swing.KeyStroke documentation for more information.

keyStrokeWithKeyCodeAndShiftModifier

```
public static javax.swing.KeyStroke keyStrokeWithKeyCodeAndShiftModifier(int keyCode)
```

Returns a KeyStroke given the numerical key code *keyCode* with the SHIFT modifier. This method also adds the appropriate modifier for the client operating system (usually CTRL_MASK). See Sun's javax.swing.KeyStroke documentation for more information.

Draft. © Apple Computer, Inc.

keyStrokeWithString

```
public static javax.swing.KeyStroke keyStrokeWithString(String keyStrokeDescription)
```

Returns a KeyStroke for the String *keyStrokeDescription*. This method adds the appropriate modifier for the client operating system (usually CTRL_MASK). See Sun's javax.swing.KeyStroke documentation for more information.

mergedActions

```
public static NSArray mergedActions(
   NSArray actionArray1,
   NSArray actionArray2)
```

Returns an NSArray containing all of the actions in *actionArray1* and *actionArray2* with duplicate actions removed.

sortedActions

```
public static NSArray sortedActions(NSArray actionArray)
```

Returns a sorted NSArray containing the actions in *actionArray*. The actions are sorted first on the category priority, then on the action priority, and finally on the description path.

See Also: categoryPriority, actionPriority, and descriptionPath.

standardActionForFocusComponent

```
public static EOAction standardActionForFocusComponent(
   String actionName,
   javax.swing.KeyStroke menuAccelerator,
   int actionPriority)
```

Returns a shared action as specified by the arguments. When the action triggers, it is dispatched to the focus component (for example, a text field). The action's category priority is the edit action priority so the action is grouped with the other edit actions.

See Also: actionName, menuAccelerator, and actionPriority.

Draft. © Apple Computer, Inc.

standardActivatePreviousWindowActionForApplication

```
public static EOAction standardActivatePreviousWindowActionForApplication()
```

Returns a shared action (an EOAction object) for the `activatePreviousWindow` method. When this action triggers, it is dispatched to the EOApplication at the top of the controller hierarchy. The action's category priority is the window action priority so the action is grouped with the other window actions. This action appears as the Activate Previous Window item in the Window menu in Direct to Java Client applications.

standardAddActionForControllerHierarchy

```
public static EOAction standardAddActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `add` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardAppendActionForControllerHierarchy

```
public static EOAction standardAppendActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `append` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the edit action priority so the action is grouped with the other edit actions.

standardCancelActionForControllerHierarchy

```
public static EOAction standardCancelActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `cancel` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the modal dialog action priority so the action is grouped with the other modal dialog actions.

Draft. © Apple Computer, Inc.

standardClearActionForControllerHierarchy

```
public static EOAction standardClearActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `clear` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the edit action priority so the action is grouped with the other edit actions.

standardCloseWindowActionForControllerHierarchy

```
public static EOAction standardCloseWindowActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `close` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the window action priority so the action is grouped with the other window actions.

standardDeleteActionForControllerHierarchy

```
public static EOAction standardDeleteActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `delete` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardDeselectActionForControllerHierarchy

```
public static EOAction standardDeselectActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `deselect` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

Draft. © Apple Computer, Inc.

standardDocumentActionForApplication

```
public static EOAction standardDocumentActionForApplication(
   String actionName,
   javax.swing.KeyStroke menuAccelerator,
   int actionPriority)
```

Returns a shared action with the method name *actionName*, menu accelerator *menuAccelerator*, and action priority *actionPriority*. When this action triggers, it is dispatched to the EOApplication at the top of the controller hierarchy. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardDocumentActionForControllerHierarchy

```
public static EOAction standardDocumentActionForControllerHierarchy(
   String actionName,
   javax.swing.KeyStroke menuAccelerator,
   int actionPriority)
```

Returns a shared action with the method name *actionName*, menu accelerator *menuAccelerator*, and action priority *actionPriority*. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardDocumentActionForControllerHierarchy

```
public static EOAction standardDocumentActionForControllerHierarchy(
   String actionName,
   String baseTitle,
   javax.swing.KeyStroke menuAccelerator,
   int actionPriority)
```

Returns a shared action as specified by the arguments. The *baseTitle* parameter is the name of the action as it appears in the user interface and is used for both the short description and the action title. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

See Also: actionName, actionTitle, shortDescription, menuAccelerator, and actionPriority.

Draft. © Apple Computer, Inc.

standardEditActionForControllerHierarchy

```
public static EOAction standardEditActionForControllerHierarchy(
   String actionName,
   javax.swing.KeyStroke menuAccelerator,
   int actionPriority)
```

Returns a shared action as specified by the arguments. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the edit action priority so the action is grouped with the other edit actions.

See Also: `actionName`, `menuAccelerator`, and `actionPriority`.

standardEditActionsForFocusComponent

```
public static NSArray standardEditActionsForFocusComponent()
```

Returns an NSArray containing shared actions for the `cut`, `copy`, and `paste` methods. When these actions trigger, they are dispatched to the focus component. Sets the category priorities for the actions to the edit category priority so the actions are grouped with the other edit actions.

standardFindActionForControllerHierarchy

```
public static EOAction standardFindActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `find` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the edit action priority so the action is grouped with the other edit actions.

standardInsertActionForControllerHierarchy

```
public static EOAction standardInsertActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `insertWithTask` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

Draft. © Apple Computer, Inc.

standardInsertWithTaskActionForControllerHierarchy

```
public static EOAction standardInsertWithTaskActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `insertWithTask` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardOkActionForControllerHierarchy

```
public static EOAction standardOkActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for an OK button in a modal dialog box. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the modal dialog action priority so the action is grouped with the other modal dialog actions.

standardOkAndSaveActionForControllerHierarchy

```
public static EOAction standardOkAndSaveActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for an OK and Save button in a modal dialog box. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the modal dialog action priority so the action is grouped with the other modal dialog actions.

standardOpenWithTaskActionForControllerHierarchy

```
public static EOAction standardOpenWithTaskActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `openWithTask` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

Draft. © Apple Computer, Inc.

standardQuitActionForApplication

```
public static EOAction standardQuitActionForApplication()
```

Returns a shared action (an EOAction object) for the `quit` method. When this action triggers, it is dispatched to the EOApplication at the top of the controller hierarchy. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardRedoActionForControllerHierarchy

```
public static EOAction standardRedoActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `redo` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the edit action priority so the action is grouped with the other edit actions.

standardRefreshActionForApplication

```
public static EOAction standardRefreshActionForApplication()
```

Returns a shared action (an EOAction object) for the `refresh` method. When this action triggers, it is dispatched to the EOApplication at the top of the controller hierarchy. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardRemoveActionForControllerHierarchy

```
public static EOAction standardRemoveActionForControllerHierarchy()
```

The action's category priority is the document action priority so the action is grouped with the other document actions.Returns a shared action (an EOAction object) for the `remove` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it.

Draft. © Apple Computer, Inc.

standardRevertActionForControllerHierarchy

```
public static EOAction standardRevertActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `revert` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardSaveActionForControllerHierarchy

```
public static EOAction standardSaveActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `save` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardSaveAllActionForApplication

```
public static EOAction standardSaveAllActionForApplication()
```

Returns a shared action (an EOAction object) for the `saveAll` method. When this action triggers, it is dispatched to the EOApplication at the top of the controller hierarchy. The action's category priority is the document action priority so the action is grouped with the other document actions.

standardSelectActionForControllerHierarchy

```
public static EOAction standardSelectActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `select` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the document action priority so the action is grouped with the other document actions.

Draft. © Apple Computer, Inc.

standardUndoActionForControllerHierarchy

```
public static EOAction standardUndoActionForControllerHierarchy()
```

Returns a shared action (an EOAction object) for the `undo` method. When the action triggers, it is dispatched to the subcontrollers of the controller that displays it. The action's category priority is the edit action priority so the action is grouped with the other edit actions.

Instance Methods

actionCanBePerformedInContextOfController

```
public boolean actionCanBePerformedInContextOfController(EOController controller)
```

Returns whether or not an action can trigger, which depends on the state of controllers in the controller hierarchy. For example, a Save action for an unedited document can not trigger.

actionName

```
public String actionName()
```

Returns the name of the method that executes when the receiver triggers.

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent actionEvent)
```

This method is called when an action is triggered, that is, the user presses the action's button or selects its menu item.

Draft. © Apple Computer, Inc.

actionPriority

```
public int actionPriority()
```

Returns the receiver's action priority, which determines the order in which its button or menu item appears within a category.

See Also: `categoryPriority`

actionTitle

```
public String actionTitle()
```

Returns the receiver's title, the last component of the receiver's description path.

addPropertyChangeListener

```
public void addPropertyChangeListener(java.beans.PropertyChangeListener listener)
```

See the method description for `addPropertyChangeListener` in Sun's documentation for javax.swing.AbstractAction.

categoryPriority

```
public int categoryPriority()
```

Returns the receiver's category priority, which determines the order in which the group of buttons or menu items that contains the receiver appears.

descriptionPath

```
public String descriptionPath()
```

Returns the receiver's menu hierarchy path. For example, the Quit menu item description path is `Document/Quit`.

Draft. © Apple Computer, Inc.

descriptionPathComponents

```
public NSArray descriptionPathComponents()
```

Returns an NSArray containing the separate components of the receiver's menu hierarchy path.

equals

```
public boolean equals(Object anObject)
```

Indicates whether some object "is equal to" this one.

firePropertyChange

```
protected void firePropertyChange(
   String propertyName,
   Object oldValue,
   Object newValue)
```

See the method description for `firePropertyChange` in Sun's documentation for javax.swing.AbstractAction.

icon

```
public javax.swing.Icon icon()
```

Returns the receiver's icon.

menuAccelerator

```
public javax.swing.KeyStroke menuAccelerator()
```

Returns the KeyStroke the user can type to invoke the receiver instead of selecting it from the menu.

Draft. © Apple Computer, Inc.

removePropertyChangeListener

```
public void removePropertyChangeListener(java.beans.PropertyChangeListener listener)
```

See the method description for `removePropertyChangeListener` in Sun's documentation for javax.swing.AbstractAction.

setActionName

```
public void setActionName(String actionName)
```

Sets the name of the method that executes when the receiver triggers.

setActionPriority

```
public void setActionPriority(int actionPriority)
```

Sets the receiver's action priority, which determines the order in which its button or menu item appears within a category.

setCategoryPriority

```
public void setCategoryPriority(int categoryPriority)
```

Returns the receiver's category priority, which determines the order in which the group of buttons or menu items containing the receiver appears.

setDescriptionPath

```
public void setDescriptionPath(String descriptionPath)
```

Sets the receiver's menu hierarchy path to *descriptionPath*.

setIcon

```
public void setIcon(javax.swing.Icon icon)
```

Sets the receiver's icon to *icon*.

Draft. © Apple Computer, Inc.

setMenuAccelerator

```
public void setMenuAccelerator(javax.swing.KeyStroke menuAccelerator)
```

Sets the KeyStroke the user can type to invoke the receiver instead of selecting it from a menu.

See Also: keyStrokeWithKeyCode, keyStrokeWithKeyCodeAndModifiers, keyStrokeWithKeyCodeAndShiftModifier, and keyStrokeWithString.

setShortDescription

```
public void setShortDescription(String shortDescription)
```

Sets the action's short description to *shortDescription*. The short description appears in buttons and menu items. If *shortDescription* is null, the receiver's title is displayed instead.

See Also: actionTitle

setSmallIcon

```
public void setSmallIcon(javax.swing.Icon anIcon)
```

Sets the receiver's small icon used for some small buttons in the user interface (the Select button in a Form window's to-one relationship editor is an example).

shortDescription

```
public String shortDescription()
```

Returns the receiver's short description, which is displayed in buttons and menu items. If the short description is set to null or has not been assigned, shortDescription returns the action's title.

See Also: actionTitle

Draft. © Apple Computer, Inc.

smallIcon

```
public javax.swing.Icon smallIcon()
```

Returns the receiver's small icon used for some small buttons in the user interface (the Select button in a Form window's to-one relationship editor is an example). By default, the small icon is not displayed for such buttons; the short description is displayed instead.

See Also: `shortDescription`

toString

```
public String toString()
```

Returns the receiver as a string that states the receiver's method name, description path, category priority, and action priority.

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOActionButtonsController

Inherits from: EOActionWidgetController :
EOComponentController :
EOController :
Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
ACTIONBUTTONSCONTROLLER	actionWidgetController

Method Types

All methods

EOActionButtonsController

actionWidget

Draft. © Apple Computer, Inc.

actionWidgetToSubcontrollerAreaDistance

createWidgetForActionsAndPlaceInContainer

disposeActionWidget

setUsesLargeButtonRepresentation

updateActionWidgetEnabling

usesLargeButtonRepresentation

Constructors

EOActionButtonsController

public EOActionButtonsController(EOXMLUnarchiver *anEOXMLUnarchiver*)

Instance Methods

actionWidget

public javax.swing.JComponent actionWidget()

actionWidgetToSubcontrollerAreaDistance

protected int actionWidgetToSubcontrollerAreaDistance()

Draft. © Apple Computer, Inc.

createWidgetForActionsAndPlaceInContainer

```
protected void createWidgetForActionsAndPlaceInContainer(
    NSArray aNSArray,
    javax.swing.JComponent aJComponent,
    int anInt)
```

disposeActionWidget

```
protected void disposeActionWidget()
```

setUsesLargeButtonRepresentation

```
public void setUsesLargeButtonRepresentation(boolean aBoolean)
```

updateActionWidgetEnabling

```
protected void updateActionWidgetEnabling()
```

usesLargeButtonRepresentation

```
public boolean usesLargeButtonRepresentation()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOActionMenuController

Inherits from: EOActionWidgetController :
EOComponentController :
EOController

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
ACTIONMENUCONTROLLER	actionWidgetController

Method Types

All methods

EOActionMenuController

actionWidget

Draft. © Apple Computer, Inc.

actionWidgetToSubcontrollerAreaDistance

createWidgetForActionsAndPlaceInContainer

disposeActionWidget

updateActionWidgetEnabling

Constructors

EOActionMenuController

public EOActionMenuController(EOXMLUnarchiver *anEOXMLUnarchiver*)

Instance Methods

actionWidget

public javax.swing.JComponent actionWidget()

actionWidgetToSubcontrollerAreaDistance

protected int actionWidgetToSubcontrollerAreaDistance()

Draft. © Apple Computer, Inc.

createWidgetForActionsAndPlaceInContainer

```
protected void createWidgetForActionsAndPlaceInContainer(
    NSArray aNSArray,
    javax.swing.JComponent aJComponent,
    int anInt)
```

disposeActionWidget

```
protected void disposeActionWidget()
```

updateActionWidgetEnabling

```
protected void updateActionWidgetEnabling()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOActionWidgetController

Inherits from: EOComponentController : EOController : Object

Implements: EOActionWidgetController.ActionCollector

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
None (abstract class)	`actionWidgetController`

Method Types

All methods

`EOActionWidgetController`

`actionWidget`

`actionWidgetContainer`

Draft. © Apple Computer, Inc.

actionWidgetPosition

actionWidgetToSubcontrollerAreaDistance

collectedActions

componentDidBecomeVisible

createWidgetForActionsAndPlaceInContainer

dispose

disposeActionWidget

generateComponent

resetActions

setActionWidgetContainer

setActionWidgetPosition

subcontrollerActionsDidChange

subcontrollerConnectionDidChange

updateActionWidgetEnabling

Constructors

EOActionWidgetController

public EOActionWidgetController(EOXMLUnarchiver *anEOXMLUnarchiver*)

Draft. © Apple Computer, Inc.

Instance Methods

actionWidget

```
public abstract javax.swing.JComponent actionWidget()
```

actionWidgetContainer

```
public javax.swing.JComponent actionWidgetContainer()
```

actionWidgetPosition

```
public int actionWidgetPosition()
```

actionWidgetToSubcontrollerAreaDistance

```
protected abstract int actionWidgetToSubcontrollerAreaDistance()
```

collectedActions

```
public NSArray collectedActions()
```


Draft. © Apple Computer, Inc.

componentDidBecomeVisible

```
protected void componentDidBecomeVisible()
```

createWidgetForActionsAndPlaceInContainer

```
protected abstract void createWidgetForActionsAndPlaceInContainer(
   NSArray aNSArray,
   javax.swing.JComponent aJComponent,
   int anInt)
```

dispose

```
public void dispose()
```

disposeActionWidget

```
protected abstract void disposeActionWidget()
```

generateComponent

```
protected void generateComponent()
```

resetActions

```
public void resetActions()
```


Draft. © Apple Computer, Inc.

setActionWidgetContainer

```
public void setActionWidgetContainer(javax.swing.JComponent aJComponent)
```

setActionWidgetPosition

```
public void setActionWidgetPosition(int anInt)
```

subcontrollerActionsDidChange

```
public void subcontrollerActionsDidChange(EOController anEOController)
```

subcontrollerConnectionDidChange

```
public void subcontrollerConnectionDidChange(EOController anEOController)
```

updateActionWidgetEnabling

```
protected abstract void updateActionWidgetEnabling()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOActionWidgets

Inherits from: Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

```
actionMenuWithActions
buttonRowWithActions
disposeActionMenu
disposeButtonRow
disposeMenuBar
menuBarWithActions
```


Draft. © Apple Computer, Inc.

updateEnablingOfActionMenu

updateEnablingOfButtonRow

updateEnablingOfMenuBar

Static Methods

actionMenuWithActions

```
public static javax.swing.JComboBox actionMenuWithActions(
   NSArray aNSArray,
   EOController anEOController,
   boolean aBoolean,
   String aString)
```

buttonRowWithActions

```
public static javax.swing.JComponent buttonRowWithActions(
   NSArray aNSArray,
   EOController anEOController,
   boolean aBoolean,
   boolean aBoolean,
   boolean aBoolean)
```

disposeActionMenu

```
public static void disposeActionMenu(javax.swing.JComboBox aJComboBox)
```


Draft. © Apple Computer, Inc.

disposeButtonRow

```
public static void disposeButtonRow(javax.swing.JComponent aJComponent)
```

disposeMenuBar

```
public static void disposeMenuBar(javax.swing.JMenuBar aJMenuBar)
```

menuBarWithActions

```
public static javax.swing.JMenuBar menuBarWithActions(
   NSArray aNSArray,
   EOController anEOController,
   boolean aBoolean)
```

updateEnablingOfActionMenu

```
public static void updateEnablingOfActionMenu(javax.swing.JComboBox aJComboBox)
```

updateEnablingOfButtonRow

```
public static void updateEnablingOfButtonRow(javax.swing.JComponent aJComponent)
```

updateEnablingOfMenuBar

```
public static void updateEnablingOfMenuBar(javax.swing.JMenuBar aJMenuBar)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOApplet

Inherits from: javax.swing.JApplet

Package: com.apple.client.eoapplication

Class Description

EOApplet is the default Applet class embedded in WebObjects pages containing a WOJavaClientApplet dynamic element. EOApplet's only task is to read all the application's arguments (passed as HTML parameters) and forward them to the initialization code in EOApplication. For maximum flexibility, any application specific code should be implemented in EOApplication's `finishInitialization` method rather than EOApplet's `init` method.

In the controller hierarchy, the applet is represented by an EOAppletController, which becomes a client subcontroller of the EOApplication object.

EOApplet is used in Java Client applications only; there is no equivalent class on the server.

Draft. © Apple Computer, Inc.

Instance Methods

init

```
public void init()
```

Instantiates an EOAppletController for the controller hierarchy and invokes EOApplication's `startApplication` using parameters retrieved via Applet's `getParameter`.

Draft. © Apple Computer, Inc.

EOAppletController

Inherits from: EOComponentController : EOController : Object

Package: com.apple.client.eoapplication

Class Description

This class represents an EOApplet as a controller in the controller hierarchy. When the application is running as an applet, this controller is the direct descendent of the EOApplication in the controller hierarchy. It performs the analogous function as a EOWindowController when the application is running as a Java application.

XML Tag	Default Rule System Controller Type
None (abstract class)	None

Method Types

All methods

```
EOAppletController
applet
```


Draft. © Apple Computer, Inc.

setApplet

setVisible

showInSupercontroller

Constructors

EOAppletController

```
public EOAppletController(EOApplet applet)
```

Creates an applet controller for *applet*.

Instance Methods

applet

```
public EOApplet applet()
```

Returns the receiver's applet.

setApplet

```
protected void setApplet(EOApplet applet)
```

Sets the receiver's applet to *applet*.

setVisible

```
public void setVisible(boolean flag)
```

Sets the visibility of the applet according to *flag*. Since applets can not be made invisible, this method does nothing if *visible* is `false`.

Draft. © Apple Computer, Inc.

showInSupercontroller

```
public boolean showInSupercontroller()
```

Properly integrates the content of the applet (usually a component of a EOInterfaceController).

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOApplication

Inherits from: EOController : Object

Implements: NSInlineObservable
NSDisposable
com.apple.client.EOKeyValueCodingAdditions
com.apple.client.EOKeyValueCoding
com.apple.client.NSKeyValueCoding
EOAction.Enabling

Package: com.apple.client.eoapplication

Class Description

Java Client applications typically execute from the command line (often referred to as a "Java application") or as an applet running in a browser. EOApplication insulates the developer from this distinction by serving as an execution-mode-independent repository for application-level client-side logic. The provided JApplet subclass EOApplet simply invokes EOApplication with the HTML arguments as parameters.

Each application has a window observer which keeps track of all of the windows in the application, which window is active, and whether all windows have been closed. The window observer has two notifications: `ActiveWindowChangedNotification` and `LastWindowClosedNotification`, which the `finishInitialization` method binds to the `activeWindowDidChange` and `lastWindowDidClose` methods, respectively.

Draft. © Apple Computer, Inc.

Each application also has a defaults manager, an EODefaults object, which maintains two dictionaries for application defaults: a transient dictionary whose values are forgotten when the application exits, and a persistent dictionary whose values are stored on the server. The defaults manager implements `valueForKey` to read the defaults and `setPersistentValueForKey` and `setTransientValueForKey` to store the defaults.

EOApplication is used in Java Client application only; there is no equivalent class on the server side.

XML Tag	Default Rule System Controller Type
None	None

Interfaces Implemented

NSInlineObservable

NSDisposable

- `dispose`

EOKeyValueCodingAdditions (com.apple.client.eocontrol)

EOAction.Enabling

- `canPerformActionNamed`

EOKeyValueCoding (com.apple.client.eocontrol inherited from EOKeyValueCodingAdditions)

NSKeyValueCoding (inherited from EOKeyValueCoding)

Method Types

Accessing the shared instance

- `sharedApplication`

Draft. © Apple Computer, Inc.

Entering the application

main

startApplication

Initializing and terminating the application

canQuit

finishInitialization

quitsOnLastWindowClose

setCanQuit

setQuitsOnLastWindowClose

Managing the application

arguments

defaults

languages

Managing documents

documents

documentsForGlobalID

editedDocuments

hasEditedDocuments

Managing the window observer

activeWindowDidChange

lastWindowDidClose

setWindowObserver

windowObserver

Methods inherited from Object

toString

Performing main menu operations

activatePreviousWindow

Draft. © Apple Computer, Inc.

collectChangesFromServer

defaultActions

saveAll

quit

Static Methods

main

```
public static void main(String[] args[])
```

This is the standard entry point for applications started from the command line (not in an applet). The *args* array contains the application's command-line arguments (for example, `-key1 value1 -key2 value2 ...`), which are stored in a parameter dictionary (NSDictionary). The user must specify an application URL (using the `-applicationURL <application URL>` argument), the name of a distribution channel class (using the `-channelClassName <channel class name>` argument), or both depending on the specific distribution channel. If the user specifies the application URL, he can optionally specify any initial entry page other than Main.

After instantiating an EODistributionChannel on the basis of these two parameters, `main` simply invokes `startApplication`.

sharedApplication

```
public static EOApplication sharedApplication()
```

Returns the EOApplication instance initialized via the `startApplication` method, throwing an IllegalStateException if `startApplication` has not yet been invoked.

Draft. © Apple Computer, Inc.

startApplication

```
public static EOApplication startApplication(
   NSDictionary parameterDictionary,
   EOComponentController initialTopComponentController,
   boolean remoteRequestArguments)
```

Creates an EOApplication. An application can execute from the command line or as an applet.

EOApplication's parameters are specified using *parameterDictionary*. If the application is a Java application, the EOApplication's `main` method reads and parses the parameters from the command line. In addition, it sets *remoteRequestArguments* to `true`, which triggers `startApplication` to read additional parameters from the applet at the URL specified on the command line. If the application is started in an applet, all parameters are contained in the HTML.

The *initialTopComponentController* parameter specifies the top-most EOComponentController in the controller hierarchy. For applets, this controller is an EOAppletController. For command line applications, the `main` method sets *initialTopComponentController* to `null`, which causes a new EOFrameController to be instantiated and used as the top-most EOComponentController.

Instance Methods

activatePreviousWindow

```
public void activatePreviousWindow()
```

Activates the previously active window. The user can invoke this method from the Window menu.

activeWindowDidChange

```
public void activeWindowDidChange(NSNotification aNSNotification)
```

This method is invoked when the user changes the active window in the receiver (usually by clicking in an inactive window). It is invoked via a notification from the receiver's window observer.

Draft. © Apple Computer, Inc.

arguments

```
public NSDictionary arguments()
```

Returns all of the receiver's arguments in a dictionary. If the application is a Java application (and not an Applet), the arguments can come from both the command line and the applet at the URL specified on the command line.

canPerformActionNamed

```
public boolean canPerformActionNamed(String actionName)
```

Conformance to EOAction.Enabling. See the method description of `canPerformActionNamed` in the interface specification for EOAction.Enabling. An action may be disallowed if it is disabled or is an `activatePreviousWindow` action and the first window is active.

canQuit

```
public boolean canQuit()
```

Returns whether or not the receiver has a Quit item in the File submenu. Defaults to `true` if the application is run from the command line and `false` if it is started in an applet.

collectChangesFromServer

```
public void collectChangesFromServer()
```

Updates the receiver's Enterprise Objects to reflect the changes sent to the server from other client applications. By default, the application does not automatically update its objects, however, the user can update the objects manually from the Document menu in Direct to Java Client applications.

defaultActions

```
protected NSArray defaultActions()
```

Returns an NSArray containing the actions (EOAction objects) the receiver can perform.

Draft. © Apple Computer, Inc.

defaults

```
public EODefaults defaults()
```

Returns the receiver's defaults manager (an EODefaults object). If your application requires the user to log in, you should override this method so it returns `null` until the user logs in.

dispose

```
public void dispose()
```

Prepares the receiver so it is disposed when Java performs garbage collection.

documents

```
public NSArray documents()
```

Returns an NSArray containing the receiver's visible documents (EODocument objects).

documentsForGlobalID

```
public NSArray documentsForGlobalID(
   com.apple.client.eocontrol.EOGlobalID globalID,
   String entityName)
```

Returns an NSArray containing the receiver's visible documents (EODocument objects) that edit Enterprise Objects with an entity name matching *entityName* and global ID matching *globalID*.

editedDocuments

```
public NSArray editedDocuments()
```

Returns an NSArray containing the receiver's visible documents (EODocument objects) that have been edited.

Draft. © Apple Computer, Inc.

finishInitialization

```
protected void finishInitialization()
```

This method is invoked after the final event thread is guaranteed to be running. If you subclass EOApplication, use this method to initialize anything relating to the user interface or event-handling. Do not perform such initialization using EOApplication's constructor.

hasEditedDocuments

```
public boolean hasEditedDocuments()
```

Returns `true` if any of the receiver's documents has been edited. Otherwise returns `false`.

languages

```
public NSArray languages()
```

Returns an NSArray containing the language names (Strings) for which the application is localized. An example language is `English`.

lastWindowDidClose

```
public void lastWindowDidClose(NSNotification aNSNotification)
```

This method is invoked when the user closes the last window in the receiver. It is invoked as a notification from the receiver's window observer.

See Also: `quitsOnLastWindowClose`

quit

```
public void quit()
```

Causes the receiver to quit (provided `canQuit` is `true`).

See Also: `canQuit`

Draft. © Apple Computer, Inc.

quitsOnLastWindowClose

```
public boolean quitsOnLastWindowClose()
```

Returns whether or not the receiver quits when the user closes all of its windows. Defaults to `true`.

saveAll

```
public boolean saveAll()
```

Attempts to save all of the receiver's edited documents and returns `true` if it succeeds.

setCanQuit

```
public void setCanQuit(boolean flag)
```

Sets whether or not the application has a quit item in the File menu.

setQuitsOnLastWindowClose

```
public void setQuitsOnLastWindowClose(boolean flag)
```

Sets whether or not the receiver quits when the user closes all of its windows.

setWindowObserver

```
public void setWindowObserver(EOWindowObserver anEOWindowObserver)
```

Sets the receiver's window observer to *anEOWindowObserver*. The window observer manages the application's windows: which window is active, how many there are, etc.

toString

```
public String toString()
```

Returns the receiver as a string that contains the results of the EOController's `toString` method, the languages the receiver supports, and the status of the `canQuit` and `quitsOnLastWindowClose` flags.

Draft. © Apple Computer, Inc.

windowObserver

```
public EOWindowObserver windowObserver()
```

Returns the receiver's window observer.

Draft. © Apple Computer, Inc.

EOArchive

Inherits from: Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

`EOArchive`

`loadArchiveNamed`

`loadArchiveNamed`

`debug`

`disposableRegistry`

`namedObjects`

Draft. © Apple Computer, Inc.

Constructors

EOArchive

```
public EOArchive(
   Object anObject,
   NSDisposableRegistry aNSDisposableRegistry)
```

Static Methods

loadArchiveNamed

```
public static NSDictionary loadArchiveNamed(
   String aString,
   Object anObject,
   String aString,
   NSDisposableRegistry aNSDisposableRegistry)
```

loadArchiveNamed

```
public static boolean loadArchiveNamed(
   String aString,
   Object anObject,
   String aString)
```


Draft. © Apple Computer, Inc.

Instance Methods

debug

```
protected void debug(String aString)
```

disposableRegistry

```
public NSDisposableRegistry disposableRegistry()
```

namedObjects

```
public NSDictionary namedObjects()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOBeanSupport

Inherits from: Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

`beanBases`

`beanClassLoader`

`beanClassName`

`beanCodeBase`

`beanDocBase`

`beanReadyToRun`

Draft. © Apple Computer, Inc.

beanReadyToUse

beanSuperclassName

looksInstantiable

looksSerializable

Static Methods

beanBases

public static java.net.URL beanBases(Object *anObject*)

beanClassLoader

public static ClassLoader beanClassLoader(Object *anObject*)

beanClassName

public static String beanClassName(Object *anObject*)

beanCodeBase

public static java.net.URL beanCodeBase(Object *anObject*)

Draft. © Apple Computer, Inc.

beanDocBase

```
public static java.net.URL beanDocBase(Object anObject)
```

beanReadyToRun

```
public static Object beanReadyToRun(Object anObject)
```

beanReadyToUse

```
public static Object beanReadyToUse(Object anObject)
```

beanSuperclassName

```
public static String beanSuperclassName(Object anObject)
```

looksInstantiable

```
public static boolean looksInstantiable(Object anObject)
```

looksSerializable

```
public static boolean looksSerializable(Object anObject)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOBoxController

Inherits from: EOComponentController : EOController : Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
BOXCONTROLLER	groupingController

Method Types

All methods

- EOBoxController
- borderType
- generateComponent
- highlightsTitle

Draft. © Apple Computer, Inc.

horizontalBorder

setBorderType

setHighlightsTitle

setHorizontalBorder

setTitleColor

setTitleFont

setTitlePosition

setUsesTitledBorder

setVerticalBorder

titleColor

titleFont

titlePosition

usesTitledBorder

verticalBorder

Constructors

EOBoxController

public EOBoxController(EOXMLUnarchiver *anEOXMLUnarchiver*)

Draft. © Apple Computer, Inc.

Instance Methods

borderType

```
public int borderType()
```

generateComponent

```
protected void generateComponent()
```

highlightsTitle

```
public boolean highlightsTitle()
```

horizontalBorder

```
public int horizontalBorder()
```

setBorderType

```
public void setBorderType(int anInt)
```


Draft. © Apple Computer, Inc.

setHighlightsTitle

```
public void setHighlightsTitle(boolean aBoolean)
```

setHorizontalBorder

```
public void setHorizontalBorder(int anInt)
```

setTitleColor

```
public void setTitleColor(java.awt.Color aColor)
```

setTitleFont

```
public void setTitleFont(java.awt.Font aFont)
```

setTitlePosition

```
public void setTitlePosition(int anInt)
```

setUsesTitledBorder

```
public void setUsesTitledBorder(boolean aBoolean)
```


Draft. © Apple Computer, Inc.

setVerticalBorder

```
public void setVerticalBorder(int anInt)
```

titleColor

```
public java.awt.Color titleColor()
```

titleFont

```
public java.awt.Font titleFont()
```

titlePosition

```
public int titlePosition()
```

usesTitledBorder

```
public boolean usesTitledBorder()
```

verticalBorder

```
public int verticalBorder()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOComponentController

Inherits from:	EOController : Object
Implements:	NSInlineObservable (Inherited from EOController) NSDisposable (Inherited from EOController) com.apple.client.eocontrol.EOKeyValueCodingAdditions (Inherited from EOController) EOAction.Enabling (Inherited from EOController) com.apple.client.eocontrol.EOKeyValueCoding (Inherited from EOKeyValueCodingAdditions) com.apple.client.foundation.NSKeyValueCoding (Inherited from EOKeyValueCoding)
Package:	com.apple.client.eoapplication

Class Description

The EOComponentController class provides behavior for controllers that manage user interface components. A component controller has a **component**, that represents the user interface for the controller itself (not for its subcontrollers), a **subcontroller area** for displaying the user interfaces for its subcontrollers, and an **integration component**—a component that represents the controller when its shown in its supercontrollers user interface.

By default, a controller's integration component is simply the controller's component. In other words, a supercontroller adds its subcontrollers' components to the subcontroller area of its component. However, the integration component can be a completely separate component. For example, the integration component for a window controller is a button that, when pushed, opens the window controller's window.

Draft. © Apple Computer, Inc.

Also by default, a controller's subcontroller area is simply the controller's component. In the simplest case, a component controller doesn't have its own user interface, but only serves to display the user interfaces of its subcontrollers. For example, EOComponentController's component is simply an EOView. It puts nothing in the view except its subcontrollers' user interfaces. Thus, the subcontroller area is the controller's component—the EOView. However, the subcontroller area can be a subcomponent of the controller's component. For example, an EOBoxController's component contains a border (etched or bezel, for example) which is the box controller's user interface. Its subcontroller area is a component located inside the border. This is where the box controller displays its subcontrollers.

Managing the Component

To access a component controller's component, use the method `component`. If the component hasn't yet been created, `component` creates it by invoking `prepareComponent`. And `prepareComponent`, in turn, invokes `generateComponent` to dynamically create the component. Subclasses should override `generateComponent`.

To see if a controller's component has been created, use the method `isComponentPrepared`. Sometimes you need to know if a component has been created, because you can't configure its behavior after its creation. For example, if you want to set a component's alignment behavior, you have to set it with the EOComponentController method `setAlignsComponents` before the component controller creates its component.

Visibility

A component controller is visible when its component is visible on screen. When a controller becomes visible, it ensures that it's connected to its supercontroller. However, a controller that's connected to its supercontroller isn't necessarily visible. For example, you might connect an invisible controller when you need to prepare it with data before making it visible.

Similarly, a controller can be "shown" or "hidden" in its supercontroller without changing the controller's visibility. The method `showInSupercontroller` ensures that the receiver's integration component is displayed in its supercontroller's component. This doesn't necessarily change the visibility of the controller. For example, a tab switch controller might switch to another view, but if the switch controller isn't visible when the change occurs, the subcontroller doesn't become visible.

Draft. © Apple Computer, Inc.

Component Appearance

A component controller's component can have an icon and a label. The component can be represented in the user interface with icon only, label only, or with both icon and label. A component specifies which representation it prefers. A controller can *prefer* to be represented with an icon only, but can't require it. This is because the controller might not have an icon. If the controller prefers icon only and has an icon, then the controller is represented with the icon only. If the controller doesn't prefer icon only and has an icon, then the controller is represented with its icon and label. If the controller doesn't have an icon, the controller is represented with the label only.

A controller always has a label. If the controller's label hasn't been explicitly set, the controller derives one from its subcontrollers.

Layout

Subclasses of EOComponentController have complete control over how they lay out their subcontrollers. EOComponentController's implementation can lay out subcontrollers in a row or a column (the default). To change the layout direction, the method `setUsesHorizontalLayout`.

In addition to horizontal / vertical layout behavior, a component can align its components or not. For example, consider a controller that uses vertical layout and contains several EOTextFieldControllers. If the controller aligns components, it left aligns the text fields. The default alignment behavior aligns components by making their corresponding labels identically sized. The width of the labels is known as the **alignment width**.

To specify a component's alignment behavior, use the method `setAlignsComponents`. To set the alignment width, use `setAlignmentWidth`.

Resizing

EOComponentController implements complex resizing behavior. For example, if a controller's component changes in a way that might affect its minimum size, the controller's supercontroller is notified and the supercontroller ensures that its subcontroller area is at least as big as the minimum size required to show all its subcontrollers.

Using the default behavior, the user interface doesn't automatically shrink. EOComponentController only resizes up to meet the minimum requirements. As much as possible it resizes components to fill the available space. A component controller can specify both horizontal and vertical resizing behavior for its component to accommodate this scheme.

Draft. © Apple Computer, Inc.

Rule System and XML Description

The following tables identify the `controllerType`, XML tag, and XML attributes used by the rule system and EOXMLUnarchiver to generate a controller hierarchy. For more information, see the section "Rule System and XML Description" (page 6) in the package introduction.

Default Rule System Controller Type

`groupingController`

XML Tag

`COMPONENTCONTROLLER`

XML Attribute	Value	Description
`alignmentWidth`	integer	See "Layout" (page 91).
`alignsComponents`	"true" or "false"	See "Layout" (page 91).
`horizontallyResizable`	"true" or "false"	See "Resizing" (page 91).
`iconName`	string	The filename of the component's icon. Uses standard resource location behavior to find the icon by name. See "Component Appearance" (page 91) for more information.
`iconURL`	string	The URL from which the icon is downloaded. See "Component Appearance" (page 91) for more information.
`label`	string	See "Component Appearance" (page 91).
`minimumHeight`	integer	The minimum height of the controller's component, not including its subcontroller area.
`minimumWidth`	integer	The minimum width of the controller's component, not including its subcontroller area.

Draft. © Apple Computer, Inc.

XML Attribute	Value	Description
prefersIconOnly	“true” or “false”	See “Component Appearance” (page 91).
usesHorizontalLayout	“true” or “false”	See “Layout” (page 91).
verticallyResizable	“true” or “false”	See “Resizing” (page 91).

Draft. © Apple Computer, Inc.

Interfaces Implemented

NSInlineObservable (Inherited from EOController)

NSDisposable (Inherited from EOController)

- `dispose`

EOKeyValueCodingAdditions (Inherited from EOController)

EOAction.Enabling (Inherited from EOController)

EOKeyValueCoding (Inherited from EOKeyValueCodingAdditions)

- `handleQueryWithUnboundKey`
- `handleTakeValueForUnboundKey`
- `storedValueForKey`
- `takeStoredValueForKey`
- `unableToSetNullForKey`

NSKeyValueCoding (Inherited from EOKeyValueCoding)

Method Types

Constructors

- `EOComponentController`

Managing the component

- `generateComponent`
- `prepareComponent`
- `setComponent`

Draft. © Apple Computer, Inc.

component

isComponentPrepared

Managing the integration component

integration ComponentDidBecomeInvisible

integrationComponentDidBecomeVisible

integrationComponent

Managing the subcontroller area

setSubcontrollerArea

subcontrollerArea

addComponentOfSubcontroller

removeComponentOfSubcontroller

Managing component visibility

showInSupercontroller

makeVisible

componentDidBecomeVisible

showSubcontroller

hideInSupercontroller

makeInvisible

componentDidBecomeInvisible

hideSubcontroller

setVisible

isVisible

Setting component appearance

setPrefersIconOnly

prefersIconOnly

setIcon

icon

Draft. © Apple Computer, Inc.

setLabel

label

Layout behavior

setUsesHorizontalLayout

usesHorizontalLayout

setAlignsComponents

alignsComponents

setAlignmentWidth

alignmentWidth

Resizing behavior

setCanResizeHorizontally

canResizeHorizontally

setCanResizeVertically

canResizeVertically

Configuring user interface sizes

setDefaultComponentSize

defaultComponentSize

ensureMinimumComponentSizeWithoutSubcontrollers

ensureMinimumSubcontrollerAreaSize

subcontrollerMinimumSizeDidChange

minimumComponentSize

minimumComponentSizeWithoutSubcontrollers

minimumIntegrationComponentSize

minimumSubcontrollerAreaSize

Determining the root component controller

isRootComponentController

Draft. © Apple Computer, Inc.

Methods inherited from EOController

canBeTransient

removeTransientSubcontroller

subcontrollerWasAdded

subcontrollerWasRemoved

Methods inherited from Object

toString

Constructors

EOComponentController

```
public EOController()
```

```
public EOComponentController(EOXMLUnarchiver unarchiver)
```

Creates a new component controller. For information on how these constructors are used and on what they do, see the method description for the `EOController` constructors in the EOController class specification.

Instance Methods

addComponentOfSubcontroller

```
protected void addComponentOfSubcontroller(EOComponentController controller)
```

Adds the integration component for the receiver's subcontroller, *controller*, to the user interface for the receiver.

Draft. © Apple Computer, Inc.

alignmentWidth

```
public int alignmentWidth()
```

Returns the receiver's alignment width.

See Also: "Layout" (page 91)

alignsComponents

```
public boolean alignsComponents()
```

Returns `true` if the receiver aligns its components, `false` otherwise.

See Also: "Layout" (page 91)

canBeTransient

```
public boolean canBeTransient()
```

Returns `true` if the controller can be transient, `false` otherwise. By default, a component controller is transient only if it's an instance of EOComponentController, not an instance of a subclass.

See Also: `canBeTransient` **(EOController)**

canResizeHorizontally

```
public boolean canResizeHorizontally()
```

Returns `true` if the receiver can resize its component horizontally, or `false` otherwise.

See Also: "Resizing" (page 91)

canResizeVertically

```
public boolean canResizeVertically()
```

Returns `true` if the receiver can resize its component vertically, or `false` otherwise.

See Also: "Resizing" (page 91)

Draft. © Apple Computer, Inc.

component

```
public javax.swing.JComponent component()
```

Returns the receiver's component, creating and preparing it first if it doesn't already exist.

See Also: "Managing the Component" (page 90), `prepareComponent`, `generateComponent`

componentDidBecomeInvisible

```
protected void componentDidBecomeInvisible()
```

Invoked by the receiver's supercontroller when the receiver's component becomes invisible, giving the receiver a chance to respond. EOComponentController's implementation invokes `breakConnection` to break the receiver's connection to the controller hierarchy.

componentDidBecomeVisible

```
protected void componentDidBecomeVisible()
```

Invoked by the receiver's supercontroller when the receiver's component becomes visible, giving the receiver a chance to respond. EOComponentController's implementation invokes `establishConnection` to ensure the receiver is connected to the controller hierarchy.

defaultComponentSize

```
public java.awt.Dimension defaultComponentSize()
```

Returns the default size for the receiver's component. This is the size the component is set to when it's created.

See Also: "Resizing" (page 91)

dispose

```
public void dispose()
```

Conformance to NSDisposable. See the method description of `dispose` in the interface specification for NSDisposable.

Draft. © Apple Computer, Inc.

ensureMinimumComponentSizeWithoutSubcontrollers

```
public void ensureMinimumComponentSizeWithoutSubcontrollers(
   int width,
   int height)
```

Ensures that the size of the receiver's component, not including the subcontroller area, is at least as large as the area specified by *width* and *height*. If it isn't, the receiver resizes its component to *width* and *height*. This method is invoked by the receiver itself when its component is changed in a way that might affect the component's minimum size. For example, suppose a label is changed and requires a larger space.

See Also: "Resizing" **(page 91)**

ensureMinimumSubcontrollerAreaSize

```
public void ensureMinimumSubcontrollerAreaSize(
   int width,
   int height)
```

Ensures that the size of the receiver's subcontroller area is at least as large as the area specified by *width* and *height*. If it isn't, the receiver resizes its subcontroller area to *width* and *height*. This method is invoked when a subcontroller's component changes in a way that might affect its minimum size.

See Also: "Resizing" **(page 91)**

generateComponent

```
protected void generateComponent()
```

Creates the receiver's component, including setting up the subcontroller area. Implementations of these methods usually invoke `setComponent` and if necessary `setSubcontrollerArea`. EOComponentController creates an EOView.

See Also: "Managing the Component" **(page 90)**

Draft. © Apple Computer, Inc.

handleTakeValueForUnboundKey

```
public void handleTakeValueForUnboundKey(
   Object value,
   String key)
```

Conformance to EOKeyValueCoding. See the method description of `handleTakeValueForUnboundKey` in the interface specification for EOKeyValueCoding.

hideInSupercontroller

```
public boolean hideInSupercontroller()
```

Invokes `hideSubcontroller` on the receiver's supercontroller to hide the receiver's component if the component (or integration component) appears in the supercontroller's user interface. Returns `true` on success, `false` otherwise. If the receiver doesn't have a supercontroller, then this method simply makes the receiver invisible. For example, a window controller which is the root component controller simply closes.

This method is invoked automatically (for example, from `makeInvisible`). You should never need to invoke it yourself.

See Also: "Visibility" **(page 90)**

hideSubcontroller

```
protected boolean hideSubcontroller(EOComponentController controller)
```

Hides *`controller`*'s user interface in the interface of the receiver. Returns `true` if the subcontroller was successfully hidden, `false` otherwise. EOComponentController's implementation simply returns `false`. This is because most controllers can't hide their subcontrollers. Examples of controllers that can hide their subcontrollers are tab view controllers, which hide a subcontroller by making another subcontroller visible. Don't invoke this method directly; invoke `hideInSupercontroller` instead.

See Also: "Visibility" **(page 90)**

Draft. © Apple Computer, Inc.

icon

```
public javax.swing.Icon icon()
```

Returns the receiver's icon, or `null` if it has none.

See Also: "Component Appearance" **(page 91)**

integrationComponent

```
public javax.swing.JComponent integrationComponent()
```

Returns the component used as the integration component in the receiver's supercontroller to represent the receiver. EOComponentController returns its component by default.

See Also: "Class Description" **(page 89)**

integrationComponentDidBecomeInvisible

```
protected void integrationComponentDidBecomeInvisible()
```

Invoked by the receiver's supercontroller when the receiver's integration component becomes invisible, giving the receiver a chance to respond. EOComponentController's implementation sets the receiver's visibility to be `false`, because by default the integration component is identical to the component.

integrationComponentDidBecomeVisible

```
protected void integrationComponentDidBecomeVisible()
```

Invoked by the receiver's supercontroller when the receiver's integration component becomes visible, giving the receiver a chance to respond. EOComponentController's implementation sets the receiver's visibility to be `true`, because by default the integration component is identical to the component.

Draft. © Apple Computer, Inc.

isComponentPrepared

```
protected boolean isComponentPrepared()
```

Returns `true` if the receiver is prepared, `false` otherwise.

See Also: "Managing the Component" **(page 90)**

isRootComponentController

```
protected boolean isRootComponentController()
```

Returns `true` if the receiver is a root component controller, `false` otherwise. A component controller is the root component controller if its supercontroller is not an instance of EOComponentController.

isVisible

```
public boolean isVisible()
```

Returns `true` if the receiver is visible, `false` otherwise. A component controller is visible if its component is on the screen. Note, showing a subcontroller in its supercontroller doesn't necessarily mean that it is visible. For example, you can show a component in a tab view, but the component won't be visible unless the tab view is visible.

See Also: "Visibility" **(page 90)**

label

```
public String label()
```

Returns the receiver's label. If the label is not explicitly set, EOComponentController's implementation attempts to derive a label from it's subcontrollers.

See Also: "Component Appearance" **(page 91)**

Draft. © Apple Computer, Inc.

makeInvisible

```
public boolean makeInvisible()
```

Makes the receiver's user interface invisible. If the receiver's supercontroller is a component controller, makes the receiver invisible by making the receiver's supercontroller invisible. Otherwise, invokes `hideInSupercontroller`. Returns `true` if the method succeeds in making the receiver invisible, `false` otherwise.

makeVisible

```
public boolean makeVisible()
```

Makes the receiver's user interface visible. Establishes the receiver's connection to its supercontrollers and invokes `showInSupercontroller`. If the receiver's supercontroller is a component controller, it also attempts to make the supercontroller visible. Returns `true` if the method succeeds in making the receiver visible, `false` otherwise.

See Also: "Visibility" **(page 90)**

minimumComponentSize

```
public java.awt.Dimension minimumComponentSize()
```

Returns the current minimum size required to display the receiver's component, including the size required for its subcontroller area.

See Also: "Resizing" **(page 91)**

minimumComponentSizeWithoutSubcontrollers

```
public java.awt.Dimension minimumComponentSizeWithoutSubcontrollers()
```

Returns the current minimum size required to display the receiver's component, excluding the subcontroller area.

See Also: "Resizing" **(page 91)**

Draft. © Apple Computer, Inc.

minimumIntegrationComponentSize

```
public java.awt.Dimension minimumIntegrationComponentSize()
```

Returns the minimum size required to display the receiver's integration component.

See Also: "Resizing" **(page 91)**

minimumSubcontrollerAreaSize

```
public java.awt.Dimension minimumSubcontrollerAreaSize()
```

Returns the minimum size of the subcontroller area to display the receiver's subcontrollers.

See Also: "Resizing" **(page 91)**

prefersIconOnly

```
public boolean prefersIconOnly()
```

Returns `true` if the receiver prefers to represent itself with only an icon, `false` otherwise.

See Also: "Component Appearance" **(page 91)**

prepareComponent

```
protected void prepareComponent()
```

If the receiver's component is not already prepared, it generates the component.

See Also: "Managing the Component" **(page 90)**

removeComponentOfSubcontroller

```
protected void removeComponentOfSubcontroller(EOComponentController controller)
```

Removes the user interface for the specified subcontroller, *controller*, from the receiver's user interface and informs *controller* that its integration component became invisible.

Draft. © Apple Computer, Inc.

removeTransientSubcontroller

```
protected boolean removeTransientSubcontroller(EOController controller)
```

See the method description for `removeTransientSubcontroller` in the EOController class specification.

setAlignmentWidth

```
public void setAlignmentWidth(int alignmentWidgth)
```

Sets the receiver's alignment width to *alignmentWidth*. Throws an IllegalStateException if the receiver is already prepared. In other words, you can only set the alignment width before the component is generated.

See Also: "Layout" **(page 91)**

setAlignsComponents

```
public void setAlignsComponents(boolean flag)
```

Sets according to *flag* whether the receiver aligns the components in its user interface. Throws an IllegalStateException if the receiver is already prepared. In other words, you can only set the alignment behavior before the component is generated.

See Also: "Layout" **(page 91)**

setCanResizeHorizontally

```
public void setCanResizeHorizontally(boolean flag)
```

Sets according to *flag* whether the receiver's component can resize horizontally. Throws an IllegalStateException if the receiver is already prepared. In other words, you can only set the horizontal resizing behavior before the component is generated.

See Also: "Resizing" **(page 91)**

Draft. © Apple Computer, Inc.

setCanResizeVertically

```
public void setCanResizeVertically(boolean flag)
```

Sets according to *flag* whether the receiver's component can resize vertically. Throws an IllegalStateException if the receiver is already prepared. In other words, you can only set the vertical resizing behavior before the component is generated.

See Also: "Resizing" **(page 91)**

setComponent

```
public void setComponent(java.awt.Component component)
```

Sets the receiver's component to *component*.

See Also: "Managing the Component" **(page 90)**

setDefaultComponentSize

```
public void setDefaultComponentSize(java.awt.Dimension dimension)
```

Sets the default size of the receiver's component to *dimension*.

See Also: "Resizing" **(page 91)**

setIcon

```
public void setIcon(javax.swing.Icon icon)
```

Sets the receiver's icon to *icon*.

See Also: "Component Appearance" **(page 91)**

setLabel

```
public void setLabel(String label)
```

Sets the receiver's label to *label*.

See Also: "Component Appearance" **(page 91)**

Draft. © Apple Computer, Inc.

setPrefersIconOnly

```
public void setPrefersIconOnly(boolean flag)
```

Sets according to *flag* whether the receiver prefers to represent itself with only an icon or with an icon and a label.

See Also: "Component Appearance" **(page 91)**

setSubcontrollerArea

```
public void setSubcontrollerArea(javax.swing.JComponent component)
```

Sets the component that holds the user interface for the receiver's subcontrollers to *component*.

See Also: "Class Description" **(page 89)**

setUsesHorizontalLayout

```
public void setUsesHorizontalLayout(boolean flag)
```

Sets according to *flag* whether the receiver uses horizontal layout. Throws an IllegalStateException if the receiver is already prepared. In other words, you can only set the layout direction before the component is generated.

See Also: "Layout" **(page 91)**

setVisible

```
public void setVisible(boolean flag)
```

Sets the visibility of the receiver according to *flag*. Invokes `componentDidBecomeVisible` or `componentDidBecomeInvisible` to notify the receiver that its visibility changed and to give the receiver the opportunity to respond appropriately. Also notifies the receiver's ancestors that a subcontroller's visibility has changed, giving the supercontrollers the opportunity to respond.

If *flag* is `true`, this method disposes of transient receivers after making them visible.

See Also: "Visibility" **(page 90)**

Draft. © Apple Computer, Inc.

showInSupercontroller

```
public boolean showInSupercontroller()
```

Invokes `showSubcontroller` to add the receiver's user interface to its supercontroller's receiver. Returns `true` on success, `false` otherwise. If the supercontroller is `null`, this method also makes the receiver visible.

Note: Invoking this method doesn't necessarily change the visibility of the receiver. For example, a switch controller might switch the component it displays, but if the switch controller isn't visible, the subcontroller doesn't become visible when it's shown.

This method is invoked automatically (for example, from `makeVisible`). You should never need to invoke it yourself.

See Also: "Visibility" **(page 90)**

showSubcontroller

```
protected boolean showSubcontroller(EOComponentController controller)
```

Adds *`controller`*'s user interface to the interface of the receiver. Returns `true` if the subcontroller was successfully shown, `false` otherwise. EOComponentController's implementation simply returns `true`: Since the integration components for subcontrollers are added to a controller's user interface automatically, the subcontrollers are already shown. EOTabSwitchController is an example of a subclass that overrides this method in a meaningful way. To show one subcontroller, the tab switch controller hides another.

See Also: "Visibility" **(page 90)**

subcontrollerArea

```
public javax.swing.JComponent subcontrollerArea()
```

Returns the component that holds the user interface for the receiver's subcontrollers.

See Also: "Class Description" **(page 89)**

Draft. © Apple Computer, Inc.

subcontrollerMinimumSizeDidChange

```
public void subcontrollerMinimumSizeDidChange(
   EOComponentController controller,
   javax.swing.JComponent component,
   java.awt.Dimension dimension)
```

Updates the receiver's user interface to accommodate a change to the subcontroller's minimum size. This method is invoked by subcontrollers when they change in a way that might affect their component's minimum size. A subcontroller sends this method with itself, its integration component, and its new minimum size as the arguments. The expectation is that the supercontroller will make space for the subcontroller if it needs to.

See Also: "Resizing" **(page 91)**

subcontrollerWasAdded

```
protected void subcontrollerWasAdded(EOController controller)
```

Invokes `addComponentOfSubcontroller` to add the integration component (if any) for the receiver's subcontroller, *`controller`*, to the receiver's user interface. Invoked from `addSubcontroller` to notify the receiver that its subcontroller *`controller`* has been added to the controller hierarchy.

subcontrollerWasRemoved

```
protected void subcontrollerWasRemoved(EOController controller)
```

Invokes `removeComponentOfSubcontroller` to remove the integration component (if any) for the receiver's subcontroller, *`controller`*, from the receiver's user interface. Invoked from `removeSubcontroller` to notify the receiver that its subcontroller *`controller`* has been removed from the controller hierarchy.

toString

```
public String toString()
```

Returns the receiver as a string that states the receiver's class name and type name, whether the receiver is connected, the number of subcontrollers, whether or not the receiver has been prepared, whether or not the receiver is visible, information about widget sizing and alignment behavior, and so on.

Draft. © Apple Computer, Inc.

usesHorizontalLayout

```
public boolean usesHorizontalLayout()
```

Returns `true` if the receiver uses a horizontal layout, `false` otherwise.

See Also: "Layout" **(page 91)**

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOController

Inherits from: Object

Implements: NSInlineObservable
NSDisposable
EOKeyValueCodingAdditions
EOAction.Enabling
EOKeyValueCoding (Inherited from EOKeyValueCodingAdditions)
NSKeyValueCoding (Inherited from EOKeyValueCoding)

Package: com.apple.client.eoapplication

Class Description

The EOController class defines basic behavior for controller objects that are responsible for managing and sometimes generating the user interface for the client side of a Java Client application. An application's controllers are arranged in a hierarchy, which describes the complete functionality of an application.

The controller hierarchy mirrors the hierarchy of windows and widgets that make up the client application's user interface. The root of the hierarchy is an EOApplication object. The EOApplication's subcontrollers are usually window or applet controllers, which themselves have subcontrollers.

The most significant functionality provided by the EOController class is managing the controller hierarchy (building, connecting, and traversing the hierarchy) and handling actions.

Draft. © Apple Computer, Inc.

Building the Controller Hierarchy

EOController defines methods for building the controller hierarchy. You can add and remove controllers (`addSubcontroller`, `removeFromSupercontroller`), be notified when the controller hierarchy changes (`subcontrollerWasAdded` and `subcontrollerWasRemoved`), and inquire about the relationships controllers have to one another (`subcontrollers`, `supercontroller`, `isAncestorOfController`, and `isSupercontrollerOfController`).

You might need to directly invoke the methods `addSubcontroller` and `removeFromSupercontroller` to programmatically manipulate the controller hierarchy. The base implementations of these methods are sufficient for most subclasses. They set and unset a controller's supercontroller (`setSupercontroller`) and notify that supercontroller that a subcontroller was added or removed.

If you write a custom controller and you need to do something special when a subcontroller is added to or removed from the controller hierarchy, override the methods `subcontrollerWasAdded` and `subcontrollerWasRemoved` to put your customizations there. Taking this approach, you shouldn't have to override the add and remove methods.

Traversing the Controller Hierarchy

EOController defines numerous methods for traversing the controller hierarchy, but a single method provides the basic traversal functionality. The method `controllerEnumeration` creates and returns an enumeration that includes all the descendents of a controller (not including the controller), all the ancestors of a controller (not including the controller), or a controller and its descendants. You can further restrict the controllers included in an enumeration by specifying an interface the controllers must implement in order to be included. For more information, see the EOController.Enumeration interface specification and the method description for `controllerEnumeration`.

Other methods that traverse the controller hierarchy use a controller enumeration to perform the traversal. There are methods that return controllers in an enumeration that match one or more key-value pairs. Methods that use key-value coding on the controllers in an enumeration, returning the first controller that has a specified key or returning the value for that key. Also, there's a method (`invokeMethod`) that invokes a particular method on the controllers in an enumeration.

Draft. © Apple Computer, Inc.

Connecting Controllers

A controller in the controller hierarchy can be connected to its supercontroller or not. Controllers are connected when they're performing their duties, and they are disconnected when they become idle. Generally controllers are connected only when their user interface is visible. For example, the controllers associated with a window are connected when the window is visible, and they're disconnected when the window becomes invisible.

When a controller *connects* to its supercontroller, it gets from its supercontroller whatever resources or information it needs, and it prepares itself in whatever way necessary to perform its duties (for example, setting delegates). Similarly, when a controller breaks its connection to its supercontroller, it cleans up its resources for an idle period.

The EOController class defines methods for connecting controllers. There are methods for connecting and disconnecting a controller from its supercontroller (`establishConnection` and `breakConnection`), and also methods that make connections all the way up the controller hierarchy (`establishConnectionToSupercontrollers`) and break connections all the way down (`breakConnectionToSubcontrollers`). Generally you use the latter methods that connect or disconnect an entire branch of a tree. EOController's implementations of all these methods is generally sufficient for subclasses. They set the connection status of a controller (`setConnected`), and notify the controller that its connection has been established or broken. You shouldn't have to override these methods.

If you do need to do something when a controller is connected or disconnected, you should override the methods `connectionWasEstablished` and `connectionWasBroken`. These methods are invoked automatically by `establishConnection` and `breakConnection`.

Accessing and Enabling Actions

Controllers define actions that users can perform (such as quitting the application) and they know how to respond to those actions when they're performed. EOController defines methods that manage a controllers actions.

A controller has a set of actions. It also keeps track of which of those actions are enabled and which are disabled. For performance reasons, EOController's method implementations cache some of this information. Thus, whenever you do something that changes a controller's actions (such as adding a new subcontroller or enabling or disabling an action), the caches must be reset. Most of the time they're reset automatically, but subclasses might need to explicitly reset them with the method `resetActions`.

Draft. © Apple Computer, Inc.

To specify the actions a subclass understands, override the method `defaultActions`. However, to find out what actions a controller understands, use `actions`. This method simply manages and returns a cache of the methods returned by `defaultActions`. Some implementations of a `defaultActions` method are potentially costly to invoke over and over again, because they dynamically build their collections of actions. The `actions` method is simply an optimization. EOController's implementation of `actions` should be sufficient for subclasses; you should never need to override it.

To find out what actions a controller can perform at a specific point in time, use the method `enabledActions`. This method returns only the controller's actions that aren't explicitly disabled. As with `actions`, `enabledActions` manages and returns a cache of methods, and EOController's implemenation should be sufficient for subclasses.

Transience

Some controllers are needed only to dynamically generate the user interface and don't serve any purpose after the user interface has been created and connected. For example, an EOTextFieldController creates a widget and a corresponding association and then is no longer needed. Controllers such as EOTextFieldController can be **transient**, because after their work is done, they can sometimes be removed from the controller hierarchy and disposed of (with `disposeIfTransient`). This keeps the controller hierarchy simple, which makes user interface management more efficient.

Controllers specify whether or not they can be transient by overriding the method `canBeTransient`. Some controllers can be transient sometimes and not other times, so not all implementations simply return `true` or `false`. For example, an EOTableController can be transient if the double click action is unassigned. If the action is assigned, however, the controller must listen for a double click and react when one occurs.

Subclasses that can be transient should invoke the method `disposeIfTransient` as soon as their work is done and they can be disposed of. Sometimes a controller's supercontroller doesn't allow the controller to be disposed of. For example, the EOTabSwitchComponent doesn't allow its subcontrollers to be disposed of even if they're transient.

Draft. © Apple Computer, Inc.

Rule System and XML Description

The following tables identify the `controllerType`, XML tag, and XML attributes used by the rule system and EOXMLUnarchiver to generate a controller hierarchy. For more information, see the section "Rule System and XML Description" (page 6) in the package introduction.

Default Rule System Controller Type

None

XML Tag

None

XML Attribute	Value	Description
`className`	string	Name of a class to instantiate instead of the default class.
`disabledActionNames`	array of strings	Names of actions to explicitly disable.
`typeName`	string	This is usually a textual representation of the specification used to generate the controller, for example "question = window, task = query". The type name is used by the controller factory to identify which windows are the same so that it can reuse resources. The typeName is also used by the defaults system to specify per-window defaults.

Draft. © Apple Computer, Inc.

Constants

EOController defines the following `int` constants to identify types of enumerations returned by the method `controllerEnumeration`:

Constant	Description
`SubcontrollersEnumeration`	An enumeration object that enumerates over a controller's subcontrollers, not including the controller itself.
`SupercontrollersEnumeration`	An enumeration object that enumerates over a controller's supercontrollers, not including the controller itself.
`ControllerAndSubcontrollers Enumeration`	An enumeration object that enumerates over a controller and its subcontrollers.

Draft. © Apple Computer, Inc.

Interfaces Implemented

NSInlineObservable

- observerData
- setObserverData

NSDisposable

- dispose

NSKeyValueCoding (Inherited from EOKeyValueCoding)

- takeValueForKey
- valueForKey

EOKeyValueCoding (Inherited from EOKeyValueCodingAdditions)

- handleQueryWithUnboundKey
- handleTakeValueForUnboundKey
- storedValueForKey
- takeStoredValueForKey
- unableToSetNullForKey

EOKeyValueCodingAdditions

- takeValueForKeyPath
- takeValuesFromDictionary
- valueForKeyPath
- valuesForKeys

EOAction.Enabling

- canPerformActionNamed

Draft. © Apple Computer, Inc.

Method Types

Constructors

EOController

Managing the controller hierarchy

addSubcontroller

subcontrollerWasAdded

removeFromSupercontroller

removeSubcontroller

subcontrollerWasRemoved

setSupercontroller

removeTransientSubcontroller

canBeTransient

subcontrollers

supercontroller

isAncestorOfController

isSupercontrollerOfController

Traversing the controller hierarchy

controllerEnumeration

controllersInEnumeration

controllerWithKeyValuePair

controllerWithKeyValuePairs

controllersWithKeyValuePair

controllersWithKeyValuePairs

hierarchicalControllerForKey

Draft. © Apple Computer, Inc.

hierarchicalValueForKey

invokeMethod

Connecting controllers

establishConnectionToSupercontrollers

establishConnection

connectionWasEstablished

breakConnectionToSubcontrollers

breakConnection

connectionWasBroken

setConnected

isConnected

Accessing and enabling actions

actions

defaultActions

enabledActions

actionWithName

actionNames

disableActionNamed

enableActionNamed

isActionNamedEnabled

resetActions

Reusing controllers

prepareForNewTask

Accessing the type name

typeName

setTypeName

Draft. © Apple Computer, Inc.

Accessing keys

accessInstanceVariablesDirectly

takeStoredValueForKeyPath

takeStoredValuesFromDictionary

Disposing

disposeIfTransient

disposableRegistry

Methods inherited from Object

toString

Constructors

EOController

```
public EOController()

public EOController(EOXMLUnarchiver unarchiver)
```

Creates and returns a new controller. The no argument constructor is used when you create a controller programmatically, whereas the version taking an unarchiver is used in a Direct to Java Client applications to create controllers from an XML description.

Controller subclasses should implement both constructors. Most commonly, controllers are created with the assistance of an unarchiver. For more information on this unarchiving, see the book *Getting Started with Direct to Java Client*.

Draft. © Apple Computer, Inc.

Static Methods

accessInstanceVariablesDirectly

```
public static boolean accessInstanceVariablesDirectly()
```

Returns `true` if the receiver accesses its instance variables directly or `false` otherwise.By default, controllers don't access instance variables directly and return `false`.

See Also: `accessInstanceVariablesDirectly` **(EOCustomObject)**

Instance Methods

actionNames

```
public NSArray actionNames()
```

Returns an array of strings naming the actions the controller defines and responds to.

See Also: "Accessing and Enabling Actions" **(page 115)**

actionWithName

```
public EOAction actionWithName(String actionName)
```

If the receiver has an action named *`actionName`*, this method returns that action; otherwise, the method returns `null`.

See Also: "Accessing and Enabling Actions" **(page 115)**

Draft. © Apple Computer, Inc.

actions

```
public NSArray actions()
```

Returns an array containing the receiver's actions. EOController's implementation caches the result of `defaultActions` and returns that. The cache is cleared with the method `resetActions`.

See Also: "Accessing and Enabling Actions" **(page 115)**

addSubcontroller

```
public void addSubcontroller(EOController subcontroller)
```

Adds *`controller`* as a subcontroller of the receiver and sets the receiver as *`controller`*'s supercontroller (first removing *`controller`* from it's supercontroller if it already has one). Invoke this method to programmatically add a subcontroller to the hierarchy.

EOController's implementation sets *`subcontroller`*'s supercontroller and notifies the receiver that a subcontroller was added. It does nothing if *`controller`* is a supercontroller of the receiver. The default implantation of this method should be sufficient for most subclasses; you shouldn't have to override it. If you need to do something special when a subcontroller is added, override `subcontrollerWasAdded`.

See Also: "Building the Controller Hierarchy" **(page 114)**

breakConnection

```
public void breakConnection()
```

Breaks the receiver's connection to its supercontroller. Invokes `connectionWasBroken` to give the receiver a chance to clean up, and informs all its ancestors that a subcontroller's connections have changed so the ancestors can respond appropriately. Use this method to programmatically disconnect a single controller (and not its subcontrollers).

EOController's implementation is sufficient for most subclasses, so you don't ordinarily override this method.

See Also: "Connecting Controllers" **(page 115)**

Draft. © Apple Computer, Inc.

breakConnectionToSubcontrollers

```
public void breakConnectionToSubcontrollers()
```

Breaks the connections the receiver's subcontrollers have to their subcontrollers, and then breaks the receiver's connections to its subcontrollers. This method is invoked recursively down the subcontroller chain until the receiver and all its subcontrollers are disconnected. Use this method to programmatically disconnect a branch of the controller hierarchy from a particular controller down.

EOController's implementation is sufficient for most subclasses, so you don't ordinarily override this method.

See Also: "Connecting Controllers" **(page 115)**

canBeTransient

```
public boolean canBeTransient()
```

Returns `true` if the controller can be transient, `false` otherwise. EOController's implementation returns `false`.

See Also: "Transience" **(page 116)**

canPerformActionNamed

```
public boolean canPerformActionNamed(String actionName)
```

Conformance to EOAction.Enabling. See the method description of `canPerformActionNamed` in the interface specification for EOAction.Enabling.

See Also: `isActionNamedEnabled`, "Accessing and Enabling Actions" **(page 115)**

connectionWasBroken

```
protected void connectionWasBroken()
```

Invoked from `breakConnection` to notify the receiver that its connection to its supercontroller has been broken, giving the receiver the opportunity to clean up after its become idle.

See Also: "Connecting Controllers" **(page 115)**

Draft. © Apple Computer, Inc.

connectionWasEstablished

```
protected void connectionWasEstablished()
```

Invoked from `establishConnection` to notify the receiver that its connection to the controller hierarchy has been established, giving the receiver the opportunity to prepare itself (for example, setting delegates).

See Also: "Connecting Controllers" **(page 115)**

controllerEnumeration

```
public EOController.Enumeration controllerEnumeration(
   int enumerationType,
   Class controllerInterface)
```

Returns an enumeration object of the specified type and interface. For example, invoking this method with `SubcontrollersEnumeration` as the *`enumerationType`* and with `MyControllerInterface` as the *`controllerInterface`* returns an enumeration object that returns the receiver's subcontrollers that implement the interface `MyControllerInterface`.

The *`enumerationType`* argument can be one of:

- `SubcontrollersEnumeration`
- `SupercontrollersEnumeration`
- `ControllerAndSubcontrollersEnumeration`

The *`controllerInterface`* argument can be the name of an interface or `null` to specify no interface, which returns all the controllers specified by *`enumerationType`*.

See Also: "Traversing the Controller Hierarchy" **(page 114)**, EOController.Enumeration **interface specification**

Draft. © Apple Computer, Inc.

controllersInEnumeration

```
public NSArray controllersInEnumeration(
   int enumerationType,
   Class controllerInterface)
```

Returns the controllers in an enumeration specified by *enumerationType* and *controllerInterface*.

See Also: controllerEnumeration

controllersWithKeyValuePair

```
public NSArray controllersWithKeyValuePair(
   int enumerationType,
   Class controllerInterface,
   String key,
   Object value)
```

Traverses the controller hierarchy, and returns the controllers in the hierarchy whose values for *key* match *value*. This method uses a controller enumeration specified by *enumerationType* and *controllerInterface* to find the controllers. The method tests the controllers returned by the enumeration for matches and returns them. Matches are determined with the method valueForKeyPath.

See Also: "Traversing the Controller Hierarchy" **(page 114),** controllerEnumeration

controllersWithKeyValuePairs

```
public NSArray controllersWithKeyValuePairs(
   int enumerationType,
   Class controllerInterface,
   NSDictionary keyValuePairs)
```

Traverses the controller hierarchy, and returns the controllers in the hierarchy whose key-value pairs match those specified in *keyValuePairs*. This method uses a controller enumeration specified by *enumerationType* and *controllerInterface* to find the controllers. The method tests the controllers returned by the enumeration for matches and returns them. Matches are determined with the method valueForKeyPath.

See Also: "Traversing the Controller Hierarchy" **(page 114),** controllerEnumeration

Draft. © Apple Computer, Inc.

controllerWithKeyValuePair

```
public EOController controllerWithKeyValuePair(
   int enumerationType,
   Class controllerInterface,
   String key,
   Object value)
```

Traverses the controller hierarchy, and returns the first controller in the hierarchy whose value for *key* matches *value*. This method uses a controller enumeration specified by *enumerationType* and *controllerInterface* to find the controller. The method tests the controllers returned by the enumeration for a match and returns the first that it matches. Matches are determined with the method `valueForKeyPath`.

See Also: "Traversing the Controller Hierarchy" **(page 114)**, `controllerEnumeration`

controllerWithKeyValuePairs

```
public EOController controllerWithKeyValuePairs(
   int enumerationType,
   Class controllerInterface,
   NSDictionary keyValuePairs)
```

Traverses the controller hierarchy, and returns the first controller in the hierarchy whose key-value pairs match those specified in *keyValuePairs*. This method uses a controller enumeration specified by *enumerationType* and *controllerInterface* to find the controller. The method tests the controllers returned by the enumeration for a match and returns the first that it matches. Matches are determined with the method `valueForKeyPath`.

See Also: "Traversing the Controller Hierarchy" **(page 114)**, `controllerEnumeration`

defaultActions

```
protected NSArray defaultActions()
```

Returns an array of the receiver's default actions (EOAction objects). A subclass of EOController should override this method to return the action it defines merged with the actions of its supercclass. Never invoke this method directly. Instead, invoke `actions`, which caches the results of `defaultActions` and is therefore more efficient.

See Also: "Accessing and Enabling Actions" **(page 115)**

Draft. © Apple Computer, Inc.

disableActionNamed

```
public void disableActionNamed(String actionName)
```

Disables the action specified by the name *actionName* and resets the receiver's actions.

See Also: "Accessing and Enabling Actions" **(page 115)**

disposableRegistry

```
public NSDisposableRegistry disposableRegistry()
```

Returns the receiver's disposable registry. This registry contains objects that will be disposed of together with the receiver. Subclasses can use the registry to register objects that should be disposed when their controller is disposed.

dispose

```
public void dispose()
```

Conformance to NSDisposable. See the method description of `dispose` in the interface specification for NSDisposable.

disposeIfTransient

```
protected boolean disposeIfTransient()
```

Disposes the receiver if it's transient, first removing it from its supercontroller with `removeTransientSubcontroller`. Returns `true` if the receiver is transient and has been disposed, `false` otherwise. If the receiver's supercontroller is non-`null`, this method also attempts to dispose of the supercontroller if it's transient.

See Also: "Transience" **(page 116)**, `removeTransientSubcontroller`

Draft. © Apple Computer, Inc.

enableActionNamed

```
public void enableActionNamed(String actionName)
```

Enables the action named *actionName* and resets the receiver's actions.

See Also: "Accessing and Enabling Actions" **(page 115)**

enabledActions

```
protected NSArray enabledActions()
```

Returns an array of the receiver's enabled actions—those of the receiver's EOAction objects that aren't explicitly disabled. This method caches the enabled actions to enhance performance. The cache is cleared with the method `resetActions`.

See Also: "Accessing and Enabling Actions" **(page 115)**

establishConnection

```
public void establishConnection()
```

Connects the receiver to the controller hierarchy. Invokes `connectionWasEstablished` to give the receiver a chance to ready the user interface. After connecting the receiver, this method disposes of it if it's transient and is therefore no longer needed. Use this method to programmatically connect a single controller (and not its ancestors).

EOController's implementation is sufficient for most subclasses, so you don't ordinarily override this method.

See Also: "Connecting Controllers" **(page 115)**

establishConnectionToSupercontrollers

```
public void establishConnectionToSupercontrollers()
```

Connects the receiver's supercontroller to the controller hierarchy, and then establishes the receiver's connection to the controller hierarchy. This method is invoked recursively up the supercontroller chain until the receiver and all its ancestors are connected. Use this method to programmatically prepare a branch of the controller hierarchy from a controller up to the root controller.

Draft. © Apple Computer, Inc.

EOController's implementation is sufficient for most subclasses, so you don't ordinarily override this method.

See Also: "Connecting Controllers" **(page 115)**

handleQueryWithUnboundKey

```
public Object handleQueryWithUnboundKey(String key)
```

Conformance to EOKeyValueCoding. See the method description of `handleQueryWithUnboundKey` in the interface specification for EOKeyValueCoding.

handleTakeValueForUnboundKey

```
public void handleTakeValueForUnboundKey(
   Object value,
   String key)
```

Conformance to EOKeyValueCoding. See the method description of `handleTakeValueForUnboundKey` in the interface specification for EOKeyValueCoding.

hierarchicalControllerForKey

```
public EOController hierarchicalControllerForKey(
   Class controllerInterface,
   String key)
```

Starting at the receiver, searches up the controller hierarchy for the first controller that implements *controllerInterface* and has a non-`null` value for *key*. Returns that controller or `null` if none of the controllers have a non-`null` value for *key*.

See Also: "Traversing the Controller Hierarchy" **(page 114)**, `controllerEnumeration`

Draft. © Apple Computer, Inc.

hierarchicalValueForKey

```
public Object hierarchicalValueForKey(
   Class controllerInterface,
   String key)
```

Starting at the receiver, searches up the controller hierarchy for the first controller that implements *controllerInterface* and has a non-`null` value for *key*. Returns the value or `null` if none of the controllers have a non-`null` value for *key*.

See Also: "Traversing the Controller Hierarchy" **(page 114)**, `controllerEnumeration`

invokeMethod

```
public void invokeMethod(
   int enumerationType,
   Class controllerInterface,
   String methodName,
   Class[] parameterTypes[],
   Object[] parameters[])
```

Traverses the controller hierarchy, invoking the method specified by *methodName* and *parameterTypes* on the appropriate controllers. This method uses a controller enumeration specified by *enumerationType* and *controllerInterface* to find the controllers on which to invoke the specified method. For each controller in the enumeration, this method invokes the *methodName* method with the values in *parameters* as arguments to the method.

See Also: "Traversing the Controller Hierarchy" **(page 114)**, `controllerEnumeration`

isActionNamedEnabled

```
public boolean isActionNamedEnabled(String actionName)
```

Returns `true` if the action specified by *actionName* isn't specifically disabled, `false` otherwise.

See Also: "Accessing and Enabling Actions" **(page 115)**

Draft. © Apple Computer, Inc.

isAncestorOfController

```
public boolean isAncestorOfController(EOController controller)
```

Returns `true` if *`controller`* is a subcontroller of the receiver, of the receiver's subcontrollers, or their subcontrollers, and so on; `false` otherwise.

isConnected

```
public boolean isConnected()
```

Returns `true` if the receiver is connected, `false` otherwise.

See Also: "Connecting Controllers" **(page 115)**

isSupercontrollerOfController

```
public boolean isSupercontrollerOfController(EOController controller)
```

Returns `true` if *`controller`* is an immediate subcontroller of the receiver, `false` otherwise.

observerData

```
public Object observerData()
```

Conformance to NSInlineObservable. See the method description of `observerData` in the interface specification for NSInlineObservable.

prepareForNewTask

```
public void prepareForNewTask(boolean prepareSubcontrollersForNewTask)
```

Prepares the receiver for performing a new task by resetting any data. If *`prepareSubcontrollersForNewTask`* is `true`, this method also sends `prepareForNewTask` to each of the receiver's subcontrollers. This method is invoked to prepare a branch of the controller hierarchy to be reused. Subclasses should override this method to get rid of data and perform any additional clean up.

Draft. © Apple Computer, Inc.

removeFromSupercontroller

```
public void removeFromSupercontroller()
```

Removes the receiver from its supercontroller's set of subcontrollers. Invoke this method when you need to programmatically remove a controller from the controller hierarchy.

EOController's implementation simply invokes `removeSubcontroller` on the receiver's supercontroller. This method is a convenience so you don't have to look up a controller's supercontroller. The default implementation should be sufficient for subclasses; you shouldn't have to override it.

See Also: "Building the Controller Hierarchy" **(page 114)**

removeSubcontroller

```
protected void removeSubcontroller(EOController subcontroller)
```

Removes *subcontroller* from the controller hierarchy. EOController's implementation disconnects *subcontroller* from the controller hierarchy, and invokes `subcontrollerWasRemoved` on the receiver to give the receiver a chance to respond appropriately. Never invoke this method directly; use `removeFromSupercontroller` instead. The default implementation should be sufficient for subclasses; you shouldn't have to override it. If you need to do something when a subcontroller is removed, implement `subcontrollerWasRemoved`.

See Also: "Building the Controller Hierarchy" **(page 114)**

removeTransientSubcontroller

```
protected boolean removeTransientSubcontroller(EOController subcontroller)
```

Removes *subcontroller* from the controller hierarchy if *subcontroller* can be transient and if the receiver allows it. Returns `true` if the subcontroller could be removed, `false` otherwise. This method is invoked from `disposeIfTransient`, which is invoked in various situations to remove controllers as soon as they can become transient.

See Also: "Transience" **(page 116)**

Draft. © Apple Computer, Inc.

resetActions

```
public void resetActions()
```

Destroys the receiver's cache of actions and enabled actions, and destroys the action caches of the receiver's supercontrollers. This method is generally invoked automatically when the receiver's set of actions changes or when an action's enabled state is changed, but you can invoke it yourself to clear the caches as needed. EOController's implementation of this method is sufficient for most subclasses. You shouldn't have to override it.

See Also: "Accessing and Enabling Actions" **(page 115)**

setConnected

```
protected void setConnected(boolean flag)
```

Sets the receiver's connected status according to *flag*. EOController's implementation is sufficient for most subclasses; you don't normally override this method. Nor should you ever need to invoke it; `establishConnection` and `breakConnection` set the controller's connection status automatically.

See Also: "Connecting Controllers" **(page 115)**

setObserverData

```
public void setObserverData(Object data)
```

Conformance to NSInlineObservable. See the method description of `setObserverData` in the interface specification for NSInlineObservable.

setSupercontroller

```
protected boolean setSupercontroller(EOController controller)
```

Sets the receiver's supercontroller to *controller* and resets the receiver's actions. Returns `true` on success or `false` otherwise. It fails if *controller* is unacceptable as the receiver's supercontroller. Also, *controller* can be `null` to unset the receiver's supercontroller.

Draft. © Apple Computer, Inc.

EOController's implementation is sufficient for most subclasses; you don't normally override this method. Nor should you ever need to invoke it; `addSubcontroller` sets the supercontroller automatically.

See Also: "Building the Controller Hierarchy" **(page 114)**

setTypeName

```
public void setTypeName(String typeName)
```

Sets the receiver's type name to *typeName*.

See Also: "Rule System and XML Description" **(page 117)**

storedValueForKey

```
public Object storedValueForKey(String key)
```

Conformance to EOKeyValueCoding. See the method description of `storedValueForKey` in the interface specification for EOKeyValueCoding.

subcontrollers

```
public NSArray subcontrollers()
```

Returns the receiver's immediate subcontrollers. Use `controllerEnumeration` of `controllersInEnumeration` to return all the controllers in the hierarchy under the receiver.

subcontrollerWasAdded

```
protected void subcontrollerWasAdded(EOController subcontroller)
```

Invoked from `addSubcontroller` to notify the receiver that its subcontroller *subcontroller* has been added to the controller hierarchy, giving the receiver the opportunity to prepare the subcontroller for use.

See Also: "Building the Controller Hierarchy" **(page 114)**

Draft. © Apple Computer, Inc.

subcontrollerWasRemoved

```
protected void subcontrollerWasRemoved(EOController subcontroller)
```

Invoked from `removeSubcontroller` to notify the receiver that its subcontroller *`subcontroller`* has been removed from the controller hierarchy, giving the receiver the opportunity to perform any necessary clean up.

See Also: "Building the Controller Hierarchy" **(page 114)**

supercontroller

```
public EOController supercontroller()
```

Returns the receiver's supercontroller, or `null` if the receiver has no supercontroller.

```
public EOController supercontroller(Class controllerInterface)
```

Searching from the receiver's immediate supercontroller, returns the first supercontroller that implements the interface *`controllerInterface`*. Returns `null` if the receiver has no supercontroller or if none of the supercontrollers implement *`controllerInterface`*. Returns the receiver's immediate supercontroller if controllerInterface is `null`.

takeStoredValueForKey

```
public void takeStoredValueForKey(
   Object value,
   String key)
```

Conformance to EOKeyValueCoding. See the method description of `takeStoredValueForKey` in the interface specification for EOKeyValueCoding.

takeStoredValueForKeyPath

```
public void takeStoredValueForKeyPath(
   Object value,
   String keyPath)
```

Do not use this method. It is considered private, and it will be removed in a future releases.

Draft. © Apple Computer, Inc.

takeStoredValuesFromDictionary

```
public void takeStoredValuesFromDictionary(NSDictionary dictionary)
```

Do not use this method. It is considered private, and it will be removed in a future releases.

takeValueForKey

```
public void takeValueForKey(
   Object value,
   String key)
```

Conformance to NSKeyValueCoding. See the method description of `valueForKey` in the interface specification for NSKeyValueCoding.

takeValueForKeyPath

```
public void takeValueForKeyPath(
   Object value,
   String keyPath)
```

Conformance to EOKeyValueCodingAdditions (com.apple.client.eocontrol). See the method description of `takeValueForKeyPath` in the interface specification for EOKeyValueCodingAdditions.

takeValuesFromDictionary

```
public void takeValuesFromDictionary(NSDictionary dictionary)
```

Conformance to EOKeyValueCodingAdditions (com.apple.client.eocontrol). See the method description of `takeValueFromDictionary` in the interface specification for EOKeyValueCodingAdditions.

toString

```
public String toString()
```

Returns the receiver as a string that states the receiver's class name and type name, whether the receiver is connected, and the number of subcontrollers.

Draft. © Apple Computer, Inc.

typeName

```
public String typeName()
```

Returns the receiver's type name—a string that uniquely identifies the receiver as a node in the controller hierarchy. EOController's implementation returns `null`. The type name is used to identify controllers that have the same task. It is used to configure a controller with user defaults and also to reuse controllers when possible.

See Also: "Rule System and XML Description" **(page 117)**

unableToSetNullForKey

```
public void unableToSetNullForKey(String key)
```

Conformance to EOKeyValueCoding. See the method description of `unableToSetNullForKey` in the interface specification for EOKeyValueCoding.

valueForKey

```
public Object valueForKey(String key)
```

Conformance to NSKeyValueCoding. See the method description of `valueForKey` in the interface specification for NSKeyValueCoding.

valueForKeyPath

```
public Object valueForKeyPath(String keyPath)
```

Conformance to EOKeyValueCodingAdditions (com.apple.client.eocontrol). See the method description of `valueForKeyPath` in the interface specification for EOKeyValueCodingAdditions.

valuesForKeys

```
public NSDictionary valuesForKeys(NSArray keys)
```

Conformance to EOKeyValueCodingAdditions (com.apple.client.eocontrol). See the method description of `valuesForKeys` in the interface specification for EOKeyValueCodingAdditions.

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EODefaultResourceBundle

Inherits from: java.util.ResourceBundle

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Instance Methods

getKeys

```
public java.util.Enumeration getKeys()
```

handleGetObject

```
protected Object handleGetObject(String aString)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EODefaults

Inherits from:	Object
Implements:	NSInlineObservable NSDisposable
Package:	com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

`clearAllValues`

`dispose`

`loadPersistentValues`

`observerData`

`savePersistentValues`

Draft. © Apple Computer, Inc.

setObserverData

setPersistentValueForKey

setTransientValueForKey

valueForKey

Instance Methods

clearAllValues

```
public void clearAllValues()
```

dispose

```
public void dispose()
```

loadPersistentValues

```
public void loadPersistentValues()
```

observerData

```
public Object observerData()
```


Draft. © Apple Computer, Inc.

savePersistentValues

```
public void savePersistentValues()
```

setObserverData

```
public void setObserverData(Object anObject)
```

setPersistentValueForKey

```
public void setPersistentValueForKey(
   Object anObject,
   String aString)
```

setTransientValueForKey

```
public void setTransientValueForKey(
   Object anObject,
   String aString)
```

valueForKey

```
public Object valueForKey(String aString)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EODialogController

Inherits from: EOSimpleWindowController :
EOWindowController :
EOComponentController :
EOController :
Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
DIALOGCONTROLLER	windowController

Method Types

All methods

EODialogController

Draft. © Apple Computer, Inc.

runControllerInNewDialog

generateBorderSize

newWindow

setWindowResizable

setWindowTitle

Constructors

EODialogController

```
public EODialogController(EOXMLUnarchiver anEOXMLUnarchiver)
```

Static Methods

runControllerInNewDialog

```
public static void runControllerInNewDialog(
   EOComponentController anEOComponentController,
   String aString)
```


Draft. © Apple Computer, Inc.

Instance Methods

generateBorderSize

```
protected java.awt.Dimension generateBorderSize()
```

newWindow

```
protected java.awt.Window newWindow(javax.swing.JComponent aJComponent)
```

setWindowResizable

```
protected void setWindowResizable(
   java.awt.Window aWindow,
   boolean aBoolean)
```

setWindowTitle

```
protected void setWindowTitle(
   java.awt.Window aWindow,
   String aString)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EODialogs

Inherits from: Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

`runChooseOperationDialog`

`runConfirmOperationDialog`

`runConfirmOperationDialog`

`runErrorDialog`

`runInformationDialog`

Draft. © Apple Computer, Inc.

Static Methods

runChooseOperationDialog

```
public static int runChooseOperationDialog(
   String aString,
   String aString,
   String aString,
   String aString)
```

runConfirmOperationDialog

```
public static boolean runConfirmOperationDialog(
   String aString,
   String aString,
   String aString,
   String aString)
```

runConfirmOperationDialog

```
public static boolean runConfirmOperationDialog(
   String aString,
   String aString,
   String aString)
```

runErrorDialog

```
public static void runErrorDialog(
   String aString,
   String aString)
```


Draft. © Apple Computer, Inc.

runInformationDialog

```
public static void runInformationDialog(
    String aString,
    String aString)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EODisplayUtilities

Inherits from: Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

```
activateWindow
activateWindowIfVisible
componentPrefersHorizontalResizing
componentPrefersVerticalResizing
displayLabelForString
fillTargetSizeWithUnionSize
```


Draft. © Apple Computer, Inc.

fillTargetSizeWithUnionSize

integrateTransientSubcontrollerComponentForController

localizedDisplayLabelForString

locateWindow

locateWindow

minimumComponentSizeWithIntegratedComponents

minimumComponentSizeWithIntegratedComponents

minimumSubcontrollerAreaSizeWithIntegratedComponents

relocateWindow

removeComponentFromParentContainer

tryToRemoveComponent

unionSize

unionSize

updateComponentInContainer

updateComponentInController

Static Methods

activateWindow

```
public static void activateWindow(java.awt.Window aWindow)
```

activateWindowIfVisible

```
public static boolean activateWindowIfVisible(java.awt.Window aWindow)
```


Draft. © Apple Computer, Inc.

componentPrefersHorizontalResizing

```
public static boolean
   componentPrefersHorizontalResizing(javax.swing.JComponent aJComponent)
```

componentPrefersVerticalResizing

```
public static boolean
   componentPrefersVerticalResizing(javax.swing.JComponent aJComponent)
```

displayLabelForString

```
public static String displayLabelForString(String aString)
```

fillTargetSizeWithUnionSize

```
public static void fillTargetSizeWithUnionSize(
   java.awt.Dimension aDimension,
   java.awt.Dimension aDimension,
   java.awt.Dimension aDimension)
```

fillTargetSizeWithUnionSize

```
public static boolean fillTargetSizeWithUnionSize(
   java.awt.Dimension aDimension,
   int anInt,
   int anInt)
```


Draft. © Apple Computer, Inc.

integrateTransientSubcontrollerComponentForController

```
public static void integrateTransientSubcontrollerComponentForController(
   EOComponentController anEOComponentController,
   EOComponentController anEOComponentController,
   java.awt.Dimension aDimension,
   boolean aBoolean)
```

localizedDisplayLabelForString

```
public static String localizedDisplayLabelForString(String aString)
```

locateWindow

```
public static void locateWindow(
   java.awt.Window aWindow,
   java.awt.Dimension aDimension,
   java.awt.Point aPoint)
```

locateWindow

```
public static void locateWindow(
   java.awt.Window aWindow,
   java.awt.Dimension aDimension,
   int anInt,
   int anInt)
```

minimumComponentSizeWithIntegratedComponents

```
public static java.awt.Dimension minimumComponentSizeWithIntegratedComponents(
   EOComponentController anEOComponentController,
   java.awt.Dimension aDimension,
```


Draft. © Apple Computer, Inc.

```
    java.awt.Dimension aDimension,
    NSArray aNSArray,
    boolean aBoolean)
```

minimumComponentSizeWithIntegratedComponents

```
public static java.awt.Dimension minimumComponentSizeWithIntegratedComponents(
    EOComponentController anEOComponentController,
    java.awt.Dimension aDimension,
    java.awt.Dimension aDimension,
    NSArray aNSArray)
```

minimumSubcontrollerAreaSizeWithIntegratedComponents

```
public static java.awt.Dimension minimumSubcontrollerAreaSizeWithIntegratedComponents(
    java.awt.Dimension aDimension,
    NSArray aNSArray,
    boolean aBoolean)
```

relocateWindow

```
public static void relocateWindow(
    java.awt.Window aWindow,
    java.awt.Dimension aDimension,
    int anInt)
```

removeComponentFromParentContainer

```
public static void removeComponentFromParentContainer(java.awt.Component aComponent)
```


Draft. © Apple Computer, Inc.

tryToRemoveComponent

```
public static void tryToRemoveComponent(java.awt.Component aComponent)
```

unionSize

```
public static java.awt.Dimension unionSize(
   java.awt.Dimension aDimension,
   java.awt.Dimension aDimension)
```

unionSize

```
public static java.awt.Dimension unionSize(
   int anInt,
   int anInt,
   int anInt,
   int anInt)
```

updateComponentInContainer

```
public static void updateComponentInContainer(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   java.awt.Dimension aDimension,
   boolean aBoolean,
   boolean aBoolean,
   javax.swing.JComponent aJComponent,
   boolean aBoolean)
```

updateComponentInController

```
public static void updateComponentInController(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
```


Draft. © Apple Computer, Inc.

```
    java.awt.Dimension aDimension,
    boolean aBoolean,
    boolean aBoolean,
    boolean aBoolean,
    boolean aBoolean)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EODocumentController

Inherits from:	EOEntityController : EOComponentController : EOController
Implements:	EODocument EOEditable EOAssociationConnector (Inherited from EOEntityController) EOComponentController.EndEditing (Inherited from EOEntityController) EOObserving (Inherited from EOEntityController) EOObjectDisplay (Inherited from EOEntityController) NSInlineObservable (Inherited from EOController) NSDisposable (Inherited from EOController) EOKeyValueCodingAdditions (Inherited from EOController) EOAction.Enabling (Inherited from EOController) EOKeyValueCoding (Inherited from EOKeyValueCodingAdditions) NSKeyValueCoding (Inherited from EOKeyValueCoding)
Package:	com.apple.client.eoapplication

Class Description

The EODocumentController class provides behavior for displaying and editing enterprise objects in a user interface. EODocumentController's API is mostly specified by the interfaces EODocument and EOEditable. Additionally, much of the way that EODocumentController works is set up by its superclass, EOEntityController. Since EOEntityControllers use EOEditingContexts and EODisplayGroups to manage and display a set of enterprise objects;

Draft. © Apple Computer, Inc.

EODocumentControllers use them as well. However, in addition to displaying enterprise objects, document controllers can also edit their objects. You can insert, update, and delete enterprise objects; undo and redo unsaved changes; and save and revert.

EODocumentController provides several methods that interact with a user. For example, the methods `revert` and `saveIfUserConfirms` open dialogs to confirm that a user wants to revert or save before performing the action. Also, many of the methods open dialogs when an error occurs, telling the user what happened.

Root Document Controller Responsibilities

EODocumentController defines the concept of a root document controller. A document controller is the root if none of its ancestors are EODocuments. A root document controller usually provides the editing context for all its descendent document controllers—they typically don't have their own. Consequently, the root document controller has responsibilities that non-root document controllers don't have. For example, only the root document controller provides save and revert behavior.

Rule System and XML Description

The following tables identify the `controllerType`, XML tag, and XML attributes used by the rule system and EOXMLUnarchiver to generate a controller hierarchy. For more information, see the section "Rule System and XML Description" (page 6) in the package introduction.

Default Rule System Controller Type

`entityController`

XML Tag

`DOCUMENTCONTROLLER`

XML Attribute	Value	Description
`editability`	string	One of "Never", "Always", or "IfSupercontroller". See the `EOEditable` interface specification for more information on these settings.

Draft. © Apple Computer, Inc.

Interfaces Implemented

EODocument

- isDocumentForGlobalID
- isEdited
- save
- saveIfUserConfirms
- setEdited

EOEditable

- editability
- isEditable
- setEditability
- supercontrollerEditabilityDidChange
- takeResposibilityForEditabilityOfAssociation

EOAssociationConnector (Inherited from EOEntityController)

EOComponentController.EndEditing (Inherited from EOEntityController)

EOObserving (Inherited from EOEntityController)

EOObjectDisplay (Inherited from EOEntityController)

NSInlineObservable (Inherited from EOController)

NSDisposable (Inherited from EOController)

- dispose

EOKeyValueCodingAdditions (Inherited from EOController)

EOAction.Enabling (Inherited from EOController)

Draft. © Apple Computer, Inc.

canPerformActionNamed

EOKeyValueCoding (Inherited from EOKeyValueCodingAdditions)

NSKeyValueCoding (Inherited from EOKeyValueCoding)

Method Types

Constructors

EODocumentController

Inserting, updating, and deleting

insertObject

deleteSelectedObjects

wasEdited

Saving

canSave

saveChanges

saveAndMakeInvisible

saveIfUserConfirms

saveIfUserConfirmsAndMakeInvisible

saveIfUserConfirmsAndMakeInvisible

saveFailed

Reverting

canRevert

revert

revertAndMakeInvisible

revertChanges

revertFailed

Draft. © Apple Computer, Inc.

Undoing and Redoing

canUndo

undo

canRedo

redo

Determining the root document controller

isRootDocumentController

Methods inherited from EOEntityController

handleEditingContextNotification

Methods inherited from EOController

connectionWasEstablished

defaultActions

prepareForNewTask

Methods inherited from Object

toString

Constructors

EODocumentController

```
public EODocumentController()

public EODocumentController(EOXMLUnarchiver unarchiver)
```

Creates a new document controller. For information on how these constructors are used and on what they do, see the method description for the `EOController` constructors in the EOController class specification.

Draft. © Apple Computer, Inc.

Instance Methods

canPerformActionNamed

```
public boolean canPerformActionNamed(String actionName)
```

Conformance to EOAction.Enabling. EODocumentController's implementation uses the methods `canRedo`, `canRevert`, `canSave`, and `canUndo` to determine the enabling state of the corresponding actions.

See Also: `canPerformActionNamed` (EOAction.Enabling)

canRedo

```
protected boolean canRedo()
```

Returns `true` if the receiver can redo, `false` otherwise. A document controller can redo as long as its editing context's undo manager can redo and as long as it (or one of its subcontrollers) is editable.

canRevert

```
protected boolean canRevert()
```

Returns `true` if the receiver can revert, `false` otherwise. A document controller can revert only if it's been edited and only if it's the root document controller.

canSave

```
protected boolean canSave()
```

Returns `true` if the receiver can save, `false` otherwise. A document controller can save only if it's been edited and only if it's the root document controller.

Draft. © Apple Computer, Inc.

canUndo

```
protected boolean canUndo()
```

Returns `true` if the receiver can undo, `false` otherwise. A document controller can undo as long as its editing context's undo manager can undo and as long as it (or one of its subcontrollers) is editable.

connectionWasEstablished

```
protected void connectionWasEstablished()
```

See the method description for `connectionWasEstablished` in the EOController class specification. EODocumentController's implementation additionally updates its editability.

defaultActions

```
protected NSArray defaultActions()
```

Adds actions for handling editing to the default actions defined by the superclass, EOEntityController. More specifically, it adds save and revert actions. However, note that `defaultActions` only adds save and revert if the receiver is the root document controller, if it's editable, and if it's not modal.

deleteSelectedObjects

```
public void deleteSelectedObjects()
```

Deletes the objects selected in the receiver's display group and then sets the receiver's edited state to `true`.

dispose

```
public void dispose()
```

Conformance to NSDisposable. See the method description of `dispose` in the interface specification for NSDisposable.

Draft. © Apple Computer, Inc.

editability

```
public int editability()
```

Conformance to EOEditable. See the method description of `editability` in the interface specification for EOEditable.

handleEditingContextNotification

```
public void handleEditingContextNotification(NSNotification notification)
```

See the method description for `handleEditingContextNotification` in the EOEntityController class specification. EODocumentController's implementation additionally updates its edited state if the receiver is a root document controller.

insertObject

```
public void insertObject()
```

Creates a new enterprise object, inserts it into the receiver's display group, and sets the receiver's edited status to `true`.

isDocumentForGlobalID

```
public boolean isDocumentForGlobalID(
   com.apple.client.eocontrol.EOGlobalID globalID,
   String entityName)
```

Conformance to EODocument. See the method description of `isDocumentForGlobalID` in the interface specification for EODocument.

isEditable

```
public boolean isEditable()
```

Conformance to EOEditable. See the method description of `isEditable` in the interface specification for EOEditable.

Draft. © Apple Computer, Inc.

isEdited

```
public boolean isEdited()
```

Conformance to EODocument. See the method description of `isEdited` in the interface specification for EODocument.

isRootDocumentController

```
protected boolean isRootDocumentController()
```

Returns `true` if none of the supercontrollers are EODocuments, `false` otherwise.

prepareForNewTask

```
public void prepareForNewTask(boolean flag)
```

See the method description for `prepareForNewTask` in the EOController class specification. EODocumentController's implementation additionally sets its edited state to `false`.

redo

```
public void redo()
```

Tells the receiver's editing context to redo.

revert

```
public boolean revert()
```

Reverts the receiver's unsaved changes upon user confirmation. If the receiver has been edited, opens a dialog to verify that the user wants to revert. Upon confirmation, invokes `revertChanges` requesting an error dialog upon failure. Returns `true` on success, `false` upon failure or if the user cancels the revert.

Draft. © Apple Computer, Inc.

revertAndMakeInvisible

```
public boolean revertAndMakeInvisible()
```

Reverts the receiver's unsaved changes and makes the receiver invisible. Reverts by invoking `revertChanges`, requesting an error dialog upon failure Returns `true` if changes are successfully reverted, `false` if the receiver can't be reverted or if the revert fails.

revertChanges

```
public boolean revertChanges(boolean showErrorDialog)
```

Tells the receiver's editing context to revert, refetches if necessary, and sets the receiver's editing state to `false`. If the revert fails, catches the exception and, if *`showErrorDialog`* is true, invokes `revertFailed` to show the reason for failure. Returns `true` if the revert succeeds, `false` otherwise.

revertFailed

```
protected void revertFailed(
   Exception exception,
   boolean showErrorDialog)
```

If *`showErrorDialog`* is `true`, brings the receiver's user interface to the front and opens a dialog displaying *`exception`*'s class name and exception message. Invoked from `revertChanges`.

save

```
public boolean save()
```

Saves the receiver's changes. Saves by invoking `saveChanges`, requesting an error dialog upon failure Returns `true` if changes are successfully saved, `false` if the receiver can't save or if the save fails.

saveAndMakeInvisible

```
public boolean saveAndMakeInvisible()
```

Saves the receiver's changes and makes the receiver invisible. Saves by invoking `saveChanges`, requesting an error dialog upon failure Returns `true` if changes are successfully reverted, `false` if the receiver can't be reverted or if the revert fails.

Draft. © Apple Computer, Inc.

saveChanges

```
public boolean saveChanges(
   boolean showErrorDialog,
   String saveOperationTitle)
```

Tells the receiver's editing context to save changes and sets the receiver's editing state to `false`. If the save fails, catches the exception and, if *showErrorDialog* is true, invokes `saveFailed` to show the reason for failure. Returns `true` if the save succeeds, `false` otherwise.

saveFailed

```
protected void saveFailed(
   Exception showErrorDialog,
   boolean showErrorDialog,
   String saveOperationTitle)
```

If *showErrorDialog* is `true`, brings the receiver's user interface to the front and opens a dialog displaying *exception*'s class name and exception message. Invoked from `saveChanges`.

saveIfUserConfirms

```
public boolean saveIfUserConfirms(
   String operationTitle,
   String message)

public boolean saveIfUserConfirms()
```

Saves the receiver's unsaved changes upon user confirmation. If the receiver has been edited, opens a dialog to verify that the user wants to save. If *operationTitle* and *message* are provided, they are used as the dialog title and message; otherwise, "Save" and "Save Changes?" are used. Upon confirmation, invokes `saveChanges` requesting an error dialog upon failure. Returns `true` on success, `false` upon failure or if the user cancels the save.

Draft. © Apple Computer, Inc.

saveIfUserConfirmsAndMakeInvisible

```
public boolean saveIfUserConfirmsAndMakeInvisible(
   String operationTitle,
   String message)

public boolean saveIfUserConfirmsAndMakeInvisible()
```

Saves the receiver's unsaved changes upon user confirmation and makes the receiver invisible. Saves by invoking `saveIfUserConfirms`, requesting an error dialog upon failure. The arguments *`operationTitle`* and *`message`* are used as the title and message of the confirmation panel. "Save" and "Save changes?" are substituted for `null`. If the no-argument form of this method is invoked, then the title of the confirmation dialog is "Close" and the dialog has no message. Returns `true` if changes are successfully saved, `false` if the receiver can't be saved or if the save fails.

setEditability

```
public void setEditability(int editability)
```

Conformance to EOEditable. See the method description of `setEditability` in the interface specification for EOEditable.

setEdited

```
public void setEdited(boolean flag)
```

Conformance to EODocument. See the method description of `setEdited` in the interface specification for EODocument.

supercontrollerEditabilityDidChange

```
public void supercontrollerEditabilityDidChange()
```

Conformance to EOEditable. See the method description of `supercontrollerEditabilityDidChange` in the interface specification for EOEditable. EODocumentController's implementation updates the receiver's editability and resets its actions.

Draft. © Apple Computer, Inc.

takeResposibilityForEditabilityOfAssociation

```
public void
   takeResposibilityForEditabilityOfAssociation(com.apple.client.eointerface.EOAssociat
   ion association)
```

Conformance to EOEditable. See the method description of `takeResposibilityForEditabilityOfAssociation` in the interface specification for EOEditable.

toString

```
public String toString()
```

Returns the receiver as a string, including the receiver's editability and whether or not it has unsaved edits.

undo

```
public void undo()
```

Tells the receiver's editing context to redo.

wasEdited

```
protected void wasEdited()
```

Invoked from `setEdited` to notify the receiver that edited status has changed, giving the receiver the opportunity to respond.

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOEntityController

Inherits from:	EOComponentController : EOController : Object
Implements:	EOObserving EOObjectDisplay EOAssociationConnector EOComponentController.EndEditing NSInlineObservable (Inherited from EOController) NSDisposable (Inherited from EOController) EOKeyValueCodingAdditions (Inherited from EOController) EOAction.Enabling (Inherited from EOController) EOKeyValueCoding (Inherited from EOKeyValueCodingAdditions) NSKeyValueCoding (Inherited from EOKeyValueCoding)
Package:	com.apple.client.eoapplication

Class Description

The EOEntityController class provides behavior for displaying enterprise objects in a user interface that can optionally be loaded from an archive (a nib file). EOEntityController's most basic API is specified by the interface EOObjectDisplay, which identifies an implementation strategy that uses EOEditingContexts and EODisplayGroups to manage an entity controller's enterprise objects. An entity controller has an entity name, which identifies the kind of enterprise objects the controller works with. Additionally it has an editing context that manages the

Draft. © Apple Computer, Inc.

controller's enterprise objects, a display group that displays the enterprise objects and manages a selection, and a controller display group that connects controller methods to the user interface. For more information, see the EOObjectDisplay interface specification.

User Interface Archive

As a subclass of EOComponentController, EOEntityController manages a user interface component. However, whereas component controllers dynamically generate their components, entity controllers have the ability to load their components from an archive. An entity controller has an archive name, which specifies the archive from which to load the controller's component. If, however, a controller doesn't have an archive name, the controller can fall back on dynamically generating its component (an empty EOView).

Managing the Editing Context

As mentioned earlier, EOEntityController uses an editing context to manage its enterprise objects. By default, an entity controller attempts to get its editing context from a supercontroller. An entity controller looks up the controller hierarchy for the first EOObjectDisplay ancestor that has an editing context. If it finds one, the entity controller uses that supercontroller's editing context. If it doesn't find one, it creates one.

You can change the way an entity controller gets its editing context by specifying a **provider method** with `setEditingContextProviderMethodName`. If an entity controller has an editing context provider method, it gets its editing context by invoking that method.

The provider method name is a string, which can be a key path or the name of an arbitrary class's static method. For an example of setting the method name to a key path, consider a subclass of EOEntityController that implements the method `customizedEditingContext` to return an editing context for the controller to use. In this case, the provider method name could be set to "customizedEditingContext".

If the provider method name is the name of a static method, the format of the string is "<class name>:<static method name>". For example, suppose that you've written a subclass of EOApplication that implements a static method, `customizedEditingContextForAllControllers`, to return an editing context for all an application's controllers to share. Then the editing context provider method name for all entity controllers could be set to "CustomApplicationClass: customizedEditingContextForAllControllers".

Draft. © Apple Computer, Inc.

EOEntityController provides two methods that you can use as provider methods: `newEditingContext` and `nestedEditingContext`. The former simply creates a new editing context and is a convenience for setting the provider method. The latter attempts to create a new editing context that's nested inside an ancestor's editing context. If no ancestors provide an editing context to be a parent, `nestedEditingContext` simply creates a new editing context.

Managing the Display Group

EOEntityController uses a display group to display its enterprise objects. By default, an entity controller attempts to get its display group from a supercontroller. An entity controller looks up the controller hierarchy for the first EOObjectDisplay ancestor. If that supercontroller has the same entity name and a display group, the entity controller uses that supercontroller's display group. If it doesn't find one, it invokes `loadArchive` to see if a display group is provided in the archive. If the controller still doesn't have a display group, it simply creates one.

You can change the way an entity controller gets its display group by specifying a **provider method** with `setDisplayGroupProviderMethodName`. If an entity controller has a display group provider method, it gets its display group by invoking that method. The display group provider method name works the same way the editing context provider method name works. For more information, see "Managing the Editing Context" (page 178).

EOEntityController provides two methods that you can use as provider methods: `newDisplayGroup` and `newDisplayGroupUsingOptimisticRefresh`. The simply create new display groups and are convenience methods for setting the provider method.

Rule System and XML Description

The following tables identify the `controllerType`, XML tag, and XML attributes used by the rule system and EOXMLUnarchiver to generate a controller hierarchy. For more information, see the section "Rule System and XML Description" (page 6) in the package introduction.

Default Rule System Controller Type
`entityController`

XML Tag
`ENTITYCONTROLLER`

Draft. © Apple Computer, Inc.

XML Attribute	Value	Description
archive	string	The name of a nib file from which the controller loads its component (instead of dynamically creating it).
displayGroupProviderMethodName	string	A key path or string of the form "<class name>:<method name>" that names a method the controller uses to create its display group.
editingContextProviderMethodName	string	A key path or string of the form "<class name>:<method name>" that names a method the controller uses to create its editing context.
entity	string	Name of the controller's entity.

Draft. © Apple Computer, Inc.

Interfaces Implemented

EOObserving

- objectWillChange

EOObjectDisplay

- controllerDisplayGroup
- displayGroup
- editingContext
- entityName

EOAssociationConnector

- takeResposibilityForConnectionOfAssociation

EOComponentController.EndEditing

- endEditing

NSInlineObservable (Inherited from EOController)

NSDisposable (Inherited from EOController)

- dispose

EOKeyValueCodingAdditions (Inherited from EOController)

EOAction.Enabling (Inherited from EOController)

EOKeyValueCoding (Inherited from EOKeyValueCodingAdditions)

NSKeyValueCoding (Inherited from EOKeyValueCoding)

Method Types

Draft. © Apple Computer, Inc.

Constructors

EOEntityController

Setting the entity

setEntityName

Loading an archive

prepareComponent

loadArchive

controllerDidLoadArchive

objectForOutletPath

setArchiveName

archiveName

Managing the editing context

newEditingContext

setEditingContext

setEditingContextProviderMethodName

editingContextProviderMethodName

nestedEditingContext

startListeningToEditingContext

stopListeningToEditingContext

handleEditingContextNotification

setResetsEditingContextWhenPreparingForNewTask

resetsEditingContextWhenPreparingForNewTask

Managing the controller display group

setControllerDisplayGroup

hasControllerDisplayGroup

Managing the objects display group

newDataSource

Draft. © Apple Computer, Inc.

newDisplayGroup

newDisplayGroupUsingOptimisticRefresh

setDisplayGroup

startListeningToDisplayGroup

stopListeningToDisplayGroup

setObjectWithGlobalID

setObjectsWithFetchSpecification

displayGroupSortOrderings

setDisplayGroupProviderMethodName

displayGroupProviderMethodName

Accessing selected objects

selectedObject

selectedObjectGlobalID

selectedObjects

selectedObjectsGlobalIDs

Fetching

fetchesOnConnect

setFetchesOnConnect

Determining the root document controller

isRootEntityController

Notifying observers of change

willChange

Methods inherited from EOController

connectionWasBroken

connectionWasEstablished

establishConnection

prepareForNewTask

Draft. © Apple Computer, Inc.

Methods inherited from Object

toString

Constructors

EOEntityController

```
public EOEntityController(EOXMLUnarchiver unarchiver)
```

Instance Methods

archiveName

```
public String archiveName()
```

connectionWasBroken

```
protected void connectionWasBroken()
```

connectionWasEstablished

```
protected void connectionWasEstablished()
```


Draft. © Apple Computer, Inc.

controllerDidLoadArchive

```
protected void controllerDidLoadArchive(NSDictionary aNSDictionary)
```

controllerDisplayGroup

```
public com.apple.client.eointerface.EODisplayGroup controllerDisplayGroup()
```

displayGroup

```
public com.apple.client.eointerface.EODisplayGroup displayGroup()
```

displayGroupProviderMethodName

```
public String displayGroupProviderMethodName()
```

displayGroupSortOrderings

```
protected NSArray displayGroupSortOrderings()
```

dispose

```
public void dispose()
```


Draft. © Apple Computer, Inc.

editingContext

```
public com.apple.client.eocontrol.EOEditingContext editingContext()
```

editingContextProviderMethodName

```
public String editingContextProviderMethodName()
```

endEditing

```
public boolean endEditing()
```

entityName

```
public String entityName()
```

establishConnection

```
public void establishConnection()
```

fetchesOnConnect

```
public boolean fetchesOnConnect()
```


Draft. © Apple Computer, Inc.

handleEditingContextNotification

```
public void handleEditingContextNotification(NSNotification aNSNotification)
```

hasControllerDisplayGroup

```
public boolean hasControllerDisplayGroup()
```

isRootEntityController

```
protected boolean isRootEntityController()
```

loadArchive

```
protected boolean loadArchive()
```

nestedEditingContext

```
public com.apple.client.eocontrol.EOEditingContext nestedEditingContext()
```

newDataSource

```
protected com.apple.client.eocontrol.EODataSource newDataSource()
```


Draft. © Apple Computer, Inc.

newDisplayGroup

```
public com.apple.client.eointerface.EODisplayGroup newDisplayGroup()
```

newDisplayGroupUsingOptimisticRefresh

```
public com.apple.client.eointerface.EODisplayGroup
   newDisplayGroupUsingOptimisticRefresh()
```

newEditingContext

```
public com.apple.client.eocontrol.EOEditingContext newEditingContext()
```

objectForOutletPath

```
public Object objectForOutletPath(
   EOArchive anEOArchive,
   String aString)
```

objectWillChange

```
public void objectWillChange(Object anObject)
```

prepareComponent

```
protected void prepareComponent()
```


Draft. © Apple Computer, Inc.

prepareForNewTask

```
public void prepareForNewTask(boolean aBoolean)
```

resetsEditingContextWhenPreparingForNewTask

```
public boolean resetsEditingContextWhenPreparingForNewTask()
```

selectedObject

```
public com.apple.client.eocontrol.EOEnterpriseObject selectedObject()
```

selectedObjectGlobalID

```
public com.apple.client.eocontrol.EOGlobalID selectedObjectGlobalID()
```

selectedObjects

```
public NSArray selectedObjects()
```

selectedObjectsGlobalIDs

```
public NSArray selectedObjectsGlobalIDs()
```


Draft. © Apple Computer, Inc.

setArchiveName

```
public void setArchiveName(String aString)
```

setControllerDisplayGroup

```
public void
   setControllerDisplayGroup(com.apple.client.eointerface.EODisplayGroup anEODisplayGro
   up)
```

setDisplayGroup

```
public void
   setDisplayGroup(com.apple.client.eointerface.EODisplayGroup anEODisplayGroup)
```

setDisplayGroupProviderMethodName

```
public void setDisplayGroupProviderMethodName(String aString)
```

setEditingContext

```
public void
   setEditingContext(com.apple.client.eocontrol.EOEditingContext anEOEditingContext)
```

setEditingContextProviderMethodName

```
public void setEditingContextProviderMethodName(String aString)
```


Draft. © Apple Computer, Inc.

setEntityName

```
public void setEntityName(String aString)
```

setFetchesOnConnect

```
public void setFetchesOnConnect(boolean aBoolean)
```

setObjectWithGlobalID

```
public void setObjectWithGlobalID(com.apple.client.eocontrol.EOGlobalID anEOGlobalID)
```

setObjectsWithFetchSpecification

```
public void
   setObjectsWithFetchSpecification(com.apple.client.eocontrol.EOFetchSpecification anE
   OFetchSpecification)
```

setResetsEditingContextWhenPreparingForNewTask

```
public void setResetsEditingContextWhenPreparingForNewTask(boolean aBoolean)
```

startListeningToDisplayGroup

```
protected void startListeningToDisplayGroup()
```


Draft. © Apple Computer, Inc.

startListeningToEditingContext

```
protected void startListeningToEditingContext()
```

stopListeningToDisplayGroup

```
protected void stopListeningToDisplayGroup()
```

stopListeningToEditingContext

```
protected void stopListeningToEditingContext()
```

takeResposibilityForConnectionOfAssociation

```
public void
   takeResposibilityForConnectionOfAssociation(com.apple.client.eointerface.EOAssociati
   on anEOAssociation)
```

toString

```
public String toString()
```

willChange

```
public void willChange()
```


Draft. © Apple Computer, Inc.

EOFrameController

Inherits from: EOSimpleWindowController :
EOWindowController :
EOComponentController :
EOController :
Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
FRAMECONTROLLER	windowController

Method Types

All methods

EOFrameController

Draft. © Apple Computer, Inc.

runControllerInNewFrame

dispose

generateBorderSize

newWindow

setWindowResizable

setWindowTitle

verifyContentMinimumSize

Constructors

EOFrameController

```
public EOFrameController(EOXMLUnarchiver anEOXMLUnarchiver)
```

Static Methods

runControllerInNewFrame

```
public static void runControllerInNewFrame(
   EOComponentController anEOComponentController,
   String aString)
```


Draft. © Apple Computer, Inc.

Instance Methods

dispose

```
public void dispose()
```

generateBorderSize

```
protected java.awt.Dimension generateBorderSize()
```

newWindow

```
protected java.awt.Window newWindow(javax.swing.JComponent aJComponent)
```

setWindowResizable

```
protected void setWindowResizable(
    java.awt.Window aWindow,
    boolean aBoolean)
```

setWindowTitle

```
protected void setWindowTitle(
    java.awt.Window aWindow,
    String aString)
```


Draft. © Apple Computer, Inc.

verifyContentMinimumSize

```
protected java.awt.Dimension verifyContentMinimumSize(
   java.awt.Window aWindow,
   java.awt.Dimension aDimension)
```


Draft. © Apple Computer, Inc.

EOInspectorController

Inherits from: EOWindowController :
EOComponentController :
EOController :
Object

Implements: EOComponentController.ResetUserInterface

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
`INSPECTORCONTROLLER`	`windowController`

Method Types

All methods

`EOInspectorController`

Draft. © Apple Computer, Inc.

activateWindow

addComponentOfSubcontroller

dispose

generateBorderSize

generateComponent

inspectorIdentifier

integrationComponentDidBecomeInvisible

integrationComponentDidBecomeVisible

resetUserInterface

setInspectorIdentifier

subcontrollerMinimumSizeDidChange

Constructors

EOInspectorController

```
public EOInspectorController(EOXMLUnarchiver anEOXMLUnarchiver)
```

Instance Methods

activateWindow

```
public void activateWindow()
```


Draft. © Apple Computer, Inc.

addComponentOfSubcontroller

```
protected void
   addComponentOfSubcontroller(EOComponentController anEOComponentController)
```

dispose

```
public void dispose()
```

generateBorderSize

```
protected java.awt.Dimension generateBorderSize()
```

generateComponent

```
protected void generateComponent()
```

inspectorIdentifier

```
public String inspectorIdentifier()
```

integrationComponentDidBecomeInvisible

```
protected void integrationComponentDidBecomeInvisible()
```


Draft. © Apple Computer, Inc.

integrationComponentDidBecomeVisible

```
protected void integrationComponentDidBecomeVisible()
```

resetUserInterface

```
public void resetUserInterface()
```

setInspectorIdentifier

```
public void setInspectorIdentifier(String aString)
```

subcontrollerMinimumSizeDidChange

```
public void subcontrollerMinimumSizeDidChange(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   java.awt.Dimension aDimension)
```


Draft. © Apple Computer, Inc.

EOInterfaceController

Inherits from: EODocumentController :
EOEntityController :
EOComponentController :
EOController :
Object

Package: com.apple.client.eoapplication

Class Description

EOInterfaceController serves as a convenient base class for logic related to the interface of client-side applications. When the WebObjectsApplication wizard in Project Builder creates a new client-side interface, it adds (to the client-side subproject) an Interface Builder nib file representing this interface and a skeletal EOInterfaceController subclass defined as the nib file's root object or "owner."

In an application constructed in conformance to the Model-View-Controller paradigm, EOInterfaceController plays the role of controller. It has four special outlets (defined in the EOEntityController superclass): its `editingContext`, its `component`, its `displayGroup`, and its `controllerDisplayGroup`, all of which you can configure using Interface Builder. The object identified by component is an AWT JComponent that functions as the view, since it is the main entry point into the user interface. Because an enterprise object must always inhabit an editing context, `editingContext` and its contents serve as the "model." The `displayGroup` is an

Draft. © Apple Computer, Inc.

EODisplayGroup containing the enterprise objects manipulated by the controller's user interface (which may will involve other display groups). The `controllerDisplayGroup` is a convenience instance containing nothing but the interface controller itself.

XML Tag	Default Rule System Controller Type
`INTERFACECONTROLLER`	`entityController`

Method Types

All methods

`EOInterfaceController`

`archiveName`

`collectChangesFromServer`

`generateComponent`

Constructors

EOInterfaceController

```
public EOInterfaceController()

public
   EOInterfaceController(com.apple.client.eocontrol.EOEditingContext editingContext)
```

Initializes a new instance then attempts to load the nib file matching the class name. The one-argument and two-argument constructors allow you to specify the editing context used as the nib file's substitution editing context during the load.

Draft. © Apple Computer, Inc.

```
public EOInterfaceController(
   com.apple.client.eocontrol.EOEditingContext editingContext,
   String archiveName)
```

Initializes a new instance then attempts to load the associated nib file identified by *archiveName*. The *editingContext* argument is used as the nib file's substitution editing context during the load.

```
public EOInterfaceController(EOXMLUnarchiver unarchiver)
```

Initializes a new instance with the contents of the *unarchiver* EOXMLUnarchiver.

Instance Methods

archiveName

```
public String archiveName()
```

Returns the name of the nib file that specifies the receiver's user interface. Defaults to the receiver's class name.

collectChangesFromServer

```
public void collectChangesFromServer()
```

Updates the receiver's editing context to reflect any changes to enterprise objects pending on the server.

generateComponent

```
protected void generateComponent()
```

Since an EOInterfaceController requires a nib file, this method is overridden to raise a NSInternalInconsistencyException.

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOMenuSwitchController

Inherits from: EOSwitchController :
EOComponentController :
EOController :
Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
MENUSWITCHCONTROLLER	dividingController

Method Types

All methods

EOMenuSwitchController

addBorderComponentForControllerToDisplayComponent

Draft. © Apple Computer, Inc.

borderSize

dispose

newDisplayComponent

removeBorderComponentForControllerFromDisplayComponent

selectedBorderComponentInDisplayComponent

showBorderComponentAtIndexInDisplayComponent

Constructors

EOMenuSwitchController

```
public EOMenuSwitchController(EOXMLUnarchiver anEOXMLUnarchiver)
```

Instance Methods

addBorderComponentForControllerToDisplayComponent

```
protected void addBorderComponentForControllerToDisplayComponent(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent)
```

borderSize

```
public java.awt.Dimension borderSize()
```


Draft. © Apple Computer, Inc.

dispose

```
public void dispose()
```

newDisplayComponent

```
protected javax.swing.JComponent newDisplayComponent()
```

removeBorderComponentForControllerFromDisplayComponent

```
protected void removeBorderComponentForControllerFromDisplayComponent(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent,
   int anInt)
```

selectedBorderComponentInDisplayComponent

```
protected javax.swing.JComponent
   selectedBorderComponentInDisplayComponent(javax.swing.JComponent aJComponent)
```

showBorderComponentAtIndexInDisplayComponent

```
protected void showBorderComponentAtIndexInDisplayComponent(
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent,
   int anInt)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOModalDialogController

Inherits from: EODialogController :
EOWindowController :
EOComponentController :
EOController :
Object

Implements: EOComponentController.Modal

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
MODALDIALOGCONTROLLER	modalDialogController

Draft. © Apple Computer, Inc.

Method Types

All methods

EOModalDialogController

runControllerInNewModalDialog

activateWindow

closeWindow

isModal

newWindow

Constructors

EOModalDialogController

```
public EOModalDialogController(EOXMLUnarchiver anEOXMLUnarchiver)
```

Static Methods

runControllerInNewModalDialog

```
public static void runControllerInNewModalDialog(
   EOComponentController anEOComponentController,
   String aString)
```


Draft. © Apple Computer, Inc.

Instance Methods

activateWindow

```
public void activateWindow()
```

closeWindow

```
public boolean closeWindow()
```

isModal

```
public boolean isModal()
```

newWindow

```
protected java.awt.Window newWindow(javax.swing.JComponent aJComponent)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOProgrammaticSwitchController

Inherits from: EOSwitchController :
EOComponentController :
EOController :
Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
PROGRAMMATICSWITCHCONTROLLER	dividingController

Method Types

All methods

EOProgrammaticSwitchController

addBorderComponentForControllerToDisplayComponent

Draft. © Apple Computer, Inc.

newDisplayComponent

removeBorderComponentForControllerFromDisplayComponent

selectedBorderComponentInDisplayComponent

showBorderComponentAtIndexInDisplayComponent

Constructors

EOProgrammaticSwitchController

```
public EOProgrammaticSwitchController(EOXMLUnarchiver anEOXMLUnarchiver)
```

Instance Methods

addBorderComponentForControllerToDisplayComponent

```
protected void addBorderComponentForControllerToDisplayComponent(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent)
```

newDisplayComponent

```
protected javax.swing.JComponent newDisplayComponent()
```


Draft. © Apple Computer, Inc.

removeBorderComponentForControllerFromDisplayComponent

```
protected void removeBorderComponentForControllerFromDisplayComponent(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent,
   int anInt)
```

selectedBorderComponentInDisplayComponent

```
protected javax.swing.JComponent
   selectedBorderComponentInDisplayComponent(javax.swing.JComponent aJComponent)
```

showBorderComponentAtIndexInDisplayComponent

```
protected void showBorderComponentAtIndexInDisplayComponent(
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent,
   int anInt)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOSimpleWindowController

Inherits from:	EOWindowController : EOComponentController : EOController : Object
Implements:	WindowListener (java.awt.event package) ComponentListener (java.awt.event package) EOComponentController.ResetUserInterface
Package:	com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
None (abstract class)	`windowController`

Draft. © Apple Computer, Inc.

Method Types

All methods

EOSimpleWindowController

activateWindow

addComponentOfSubcontroller

closeWindow

componentDidBecomeInvisible

componentDidBecomeVisible

componentHidden

componentMoved

componentResized

componentShown

deactivateWindow

dispose

disposeIfDeactivated

integrationComponentDidBecomeInvisible

integrationComponentDidBecomeVisible

makeVisible

newWindow

newWindow

resetUserInterface

setDisposeIfDeactivated

setLabel

setWindow

Draft. © Apple Computer, Inc.

setWindowResizable

setWindowTitle

startListeningToWindow

stopListeningToWindow

subcontrollerEditedDidChange

subcontrollerMinimumSizeDidChange

verifyContentMinimumSize

window

windowActivated

windowClosed

windowClosing

windowDeactivated

windowDeiconified

windowIconified

windowOpened

Constructors

EOSimpleWindowController

public EOSimpleWindowController(EOXMLUnarchiver *anEOXMLUnarchiver*)

Draft. © Apple Computer, Inc.

Instance Methods

activateWindow

```
public void activateWindow()
```

addComponentOfSubcontroller

```
protected void
   addComponentOfSubcontroller(EOComponentController anEOComponentController)
```

closeWindow

```
public boolean closeWindow()
```

componentDidBecomeInvisible

```
protected void componentDidBecomeInvisible()
```

componentDidBecomeVisible

```
protected void componentDidBecomeVisible()
```


Draft. © Apple Computer, Inc.

componentHidden

```
public void componentHidden(java.awt.event.ComponentEvent aComponentEvent)
```

componentMoved

```
public void componentMoved(java.awt.event.ComponentEvent aComponentEvent)
```

componentResized

```
public void componentResized(java.awt.event.ComponentEvent aComponentEvent)
```

componentShown

```
public void componentShown(java.awt.event.ComponentEvent aComponentEvent)
```

deactivateWindow

```
public void deactivateWindow()
```

dispose

```
public void dispose()
```


Draft. © Apple Computer, Inc.

disposeIfDeactivated

```
public boolean disposeIfDeactivated()
```

integrationComponentDidBecomeInvisible

```
protected void integrationComponentDidBecomeInvisible()
```

integrationComponentDidBecomeVisible

```
protected void integrationComponentDidBecomeVisible()
```

makeVisible

```
public boolean makeVisible()
```

newWindow

```
protected abstract java.awt.Window newWindow(javax.swing.JComponent aJComponent)
```

newWindow

```
protected java.awt.Window newWindow()
```


Draft. © Apple Computer, Inc.

resetUserInterface

```
public void resetUserInterface()
```

setDisposeIfDeactivated

```
public void setDisposeIfDeactivated(boolean aBoolean)
```

setLabel

```
public void setLabel(String aString)
```

setWindow

```
public void setWindow(java.awt.Window aWindow)
```

setWindowResizable

```
protected abstract void setWindowResizable(
   java.awt.Window aWindow,
   boolean aBoolean)
```

setWindowTitle

```
protected abstract void setWindowTitle(
   java.awt.Window aWindow,
   String aString)
```


Draft. © Apple Computer, Inc.

startListeningToWindow

```
protected void startListeningToWindow()
```

stopListeningToWindow

```
protected void stopListeningToWindow()
```

subcontrollerEditedDidChange

```
public void subcontrollerEditedDidChange(EOController anEOController)
```

subcontrollerMinimumSizeDidChange

```
public void subcontrollerMinimumSizeDidChange(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   java.awt.Dimension aDimension)
```

verifyContentMinimumSize

```
protected java.awt.Dimension verifyContentMinimumSize(
   java.awt.Window aWindow,
   java.awt.Dimension aDimension)
```

window

```
public java.awt.Window window()
```


Draft. © Apple Computer, Inc.

windowActivated

```
public void windowActivated(java.awt.event.WindowEvent aWindowEvent)
```

windowClosed

```
public void windowClosed(java.awt.event.WindowEvent aWindowEvent)
```

windowClosing

```
public void windowClosing(java.awt.event.WindowEvent aWindowEvent)
```

windowDeactivated

```
public void windowDeactivated(java.awt.event.WindowEvent aWindowEvent)
```

windowDeiconified

```
public void windowDeiconified(java.awt.event.WindowEvent aWindowEvent)
```

windowIconified

```
public void windowIconified(java.awt.event.WindowEvent aWindowEvent)
```


Draft. © Apple Computer, Inc.

windowOpened

```
public void windowOpened(java.awt.event.WindowEvent aWindowEvent)
```


Draft. © Apple Computer, Inc.

EOSwitchController

Inherits from:	EOComponentController : EOController : Object
Implements:	EOComponentController.ResetUserInterface
Package:	com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
None (abstract class)	None

Method Types

All methods

```
EOSwitchController
addBorderComponentForControllerToDisplayComponent
addComponentOfSubcontroller
```


Draft. © Apple Computer, Inc.

borderComponents

borderSize

borderedSizeForComponentSize

componentDidBecomeInvisible

componentDidBecomeVisible

componentShouldChange

componentSizeForBorderedSize

componentSwitched

dispose

generateComponent

hideSubcontroller

inset

minimumComponentSize

newDisplayComponent

removeBorderComponentForControllerFromDisplayComponent

removeComponentOfSubcontroller

removeTransientSubcontroller

resetUserInterface

selectedBorderComponentInDisplayComponent

showBorderComponentAtIndex

showBorderComponentAtIndexInDisplayComponent

showSubcontroller

subcontrollerMinimumSizeDidChange

subcontrollerVisibilityDidChange

switchedControllers

visibleBorderComponentIndex

Draft. © Apple Computer, Inc.

Constructors

EOSwitchController

```
public EOSwitchController(EOXMLUnarchiver anEOXMLUnarchiver)
```

Instance Methods

addBorderComponentForControllerToDisplayComponent

```
protected abstract void addBorderComponentForControllerToDisplayComponent(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent)
```

addComponentOfSubcontroller

```
protected void
   addComponentOfSubcontroller(EOComponentController anEOComponentController)
```

borderComponents

```
protected NSArray borderComponents()
```


Draft. © Apple Computer, Inc.

borderSize

```
public java.awt.Dimension borderSize()
```

borderedSizeForComponentSize

```
public java.awt.Dimension borderedSizeForComponentSize(java.awt.Dimension aDimension)
```

componentDidBecomeInvisible

```
protected void componentDidBecomeInvisible()
```

componentDidBecomeVisible

```
protected void componentDidBecomeVisible()
```

componentShouldChange

```
public boolean componentShouldChange(int anInt)
```

componentSizeForBorderedSize

```
public java.awt.Dimension componentSizeForBorderedSize(java.awt.Dimension aDimension)
```


Draft. © Apple Computer, Inc.

componentSwitched

```
public void componentSwitched(int anInt)
```

dispose

```
public void dispose()
```

generateComponent

```
protected void generateComponent()
```

hideSubcontroller

```
protected boolean hideSubcontroller(EOComponentController anEOComponentController)
```

inset

```
protected int inset()
```

minimumComponentSize

```
public java.awt.Dimension minimumComponentSize()
```


Draft. © Apple Computer, Inc.

newDisplayComponent

```
protected abstract javax.swing.JComponent newDisplayComponent()
```

removeBorderComponentForControllerFromDisplayComponent

```
protected abstract void removeBorderComponentForControllerFromDisplayComponent(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent,
   int anInt)
```

removeComponentOfSubcontroller

```
protected void
   removeComponentOfSubcontroller(EOComponentController anEOComponentController)
```

removeTransientSubcontroller

```
protected boolean removeTransientSubcontroller(EOController anEOController)
```

resetUserInterface

```
public void resetUserInterface()
```


Draft. © Apple Computer, Inc.

selectedBorderComponentInDisplayComponent

```
protected abstract javax.swing.JComponent
   selectedBorderComponentInDisplayComponent(javax.swing.JComponent aJComponent)
```

showBorderComponentAtIndex

```
protected void showBorderComponentAtIndex(int anInt)
```

showBorderComponentAtIndexInDisplayComponent

```
protected abstract void showBorderComponentAtIndexInDisplayComponent(
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent,
   int anInt)
```

showSubcontroller

```
protected boolean showSubcontroller(EOComponentController anEOComponentController)
```

subcontrollerMinimumSizeDidChange

```
public void subcontrollerMinimumSizeDidChange(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   java.awt.Dimension aDimension)
```


Draft. © Apple Computer, Inc.

subcontrollerVisibilityDidChange

```
public void
   subcontrollerVisibilityDidChange(EOComponentController anEOComponentController)
```

switchedControllers

```
protected NSArray switchedControllers()
```

visibleBorderComponentIndex

```
protected int visibleBorderComponentIndex()
```


Draft. © Apple Computer, Inc.

EOTabSwitchController

Inherits from: EOSwitchController :
EOComponentController :
EOController :
Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
TABSWITCHCONTROLLER	dividingController

Method Types

All methods

EOTabSwitchController

addBorderComponentForControllerToDisplayComponent

Draft. © Apple Computer, Inc.

borderSize

inset

newDisplayComponent

removeBorderComponentForControllerFromDisplayComponent

selectedBorderComponentInDisplayComponent

showBorderComponentAtIndexInDisplayComponent

Constructors

EOTabSwitchController

```
public EOTabSwitchController(EOXMLUnarchiver anEOXMLUnarchiver)
```

Instance Methods

addBorderComponentForControllerToDisplayComponent

```
protected void addBorderComponentForControllerToDisplayComponent(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent)
```

borderSize

```
public java.awt.Dimension borderSize()
```


Draft. © Apple Computer, Inc.

inset

```
protected int inset()
```

newDisplayComponent

```
protected javax.swing.JComponent newDisplayComponent()
```

removeBorderComponentForControllerFromDisplayComponent

```
protected void removeBorderComponentForControllerFromDisplayComponent(
   EOComponentController anEOComponentController,
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent,
   int anInt)
```

selectedBorderComponentInDisplayComponent

```
protected javax.swing.JComponent
   selectedBorderComponentInDisplayComponent(javax.swing.JComponent aJComponent)
```

showBorderComponentAtIndexInDisplayComponent

```
protected void showBorderComponentAtIndexInDisplayComponent(
   javax.swing.JComponent aJComponent,
   javax.swing.JComponent aJComponent,
   int anInt)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOUserInterfaceParameters

Inherits from: Object

Package: com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

`actionTitleFont`

`actionTitlePosition`

`allowsActionIcons`

`allowsIcons`

`allowsSmallActionIcons`

`disabledTextBackgroundColor`

Draft. © Apple Computer, Inc.

editableTextBackgroundColor

highlightLabelColor

highlightLabelFont

highlightTitleColor

highlightTitleFont

labelColor

labelFont

largeBorder

localizedIcon

localizedString

makeIconBackgroundsTransparent

mediumBorder

minimumActionButtonSize

minimumSmallActionButtonSize

minimumSpecialActionButtonSize

optimizesMenuAccelerators

queryTextBackgroundColor

setActionTitleFont

setActionTitlePosition

setAllowsActionIcons

setAllowsIcons

setAllowsSmallActionIcons

setBorders

setDisabledTextBackgroundColor

setEditableTextBackgroundColor

setHighlightLabelColor

setHighlightLabelFont

Draft. © Apple Computer, Inc.

setHighlightTitleColor

setHighlightTitleFont

setLabelColor

setLabelFont

setMakeIconBackgroundsTransparent

setMinimumActionButtonSize

setMinimumSmallActionButtonSize

setMinimumSpecialActionButtonSize

setOptimizesMenuAccelerators

setQueryTextBackgroundColor

setSmallActionTitleFont

setSmallActionTitlePosition

setSpecialActionTitleFont

setSpecialActionTitlePosition

setStandardResourceBundle

setTitleColor

setTitleFont

setUsesSpecialColors

setUsesSpecialFonts

setUsesTitleWithActionIcons

setUsesTitleWithSmallActionIcons

setUsesWindowMenus

setWidgetFont

smallActionTitleFont

smallActionTitlePosition

smallBorder

specialActionTitleFont

Draft. © Apple Computer, Inc.

specialActionTitlePosition

standardActionIcon

standardResourceBundle

standardSmallActionIcon

titleColor

titleFont

usesSpecialColors

usesSpecialFonts

usesTitleWithActionIcons

usesTitleWithSmallActionIcons

usesWindowMenus

widgetFont

Static Methods

actionTitleFont

```
public static java.awt.Font actionTitleFont()
```

actionTitlePosition

```
public static int actionTitlePosition()
```


Draft. © Apple Computer, Inc.

allowsActionIcons

```
public static boolean allowsActionIcons()
```

allowsIcons

```
public static boolean allowsIcons()
```

allowsSmallActionIcons

```
public static boolean allowsSmallActionIcons()
```

disabledTextBackgroundColor

```
public static java.awt.Color disabledTextBackgroundColor()
```

editableTextBackgroundColor

```
public static java.awt.Color editableTextBackgroundColor()
```

highlightLabelColor

```
public static java.awt.Color highlightLabelColor()
```


Draft. © Apple Computer, Inc.

highlightLabelFont

```
public static java.awt.Font highlightLabelFont()
```

highlightTitleColor

```
public static java.awt.Color highlightTitleColor()
```

highlightTitleFont

```
public static java.awt.Font highlightTitleFont()
```

labelColor

```
public static java.awt.Color labelColor()
```

labelFont

```
public static java.awt.Font labelFont()
```

largeBorder

```
public static int largeBorder()
```


Draft. © Apple Computer, Inc.

localizedIcon

```
public static javax.swing.Icon localizedIcon(String aString)
```

localizedString

```
public static String localizedString(String aString)
```

makeIconBackgroundsTransparent

```
public static boolean makeIconBackgroundsTransparent()
```

mediumBorder

```
public static int mediumBorder()
```

minimumActionButtonSize

```
public static java.awt.Dimension minimumActionButtonSize()
```

minimumSmallActionButtonSize

```
public static java.awt.Dimension minimumSmallActionButtonSize()
```


Draft. © Apple Computer, Inc.

minimumSpecialActionButtonSize

```
public static java.awt.Dimension minimumSpecialActionButtonSize()
```

optimizesMenuAccelerators

```
public static boolean optimizesMenuAccelerators()
```

queryTextBackgroundColor

```
public static java.awt.Color queryTextBackgroundColor()
```

setActionTitleFont

```
public static void setActionTitleFont(java.awt.Font aFont)
```

setActionTitlePosition

```
public static void setActionTitlePosition(int anInt)
```

setAllowsActionIcons

```
public static void setAllowsActionIcons(boolean aBoolean)
```


Draft. © Apple Computer, Inc.

setAllowsIcons

```
public static void setAllowsIcons(boolean aBoolean)
```

setAllowsSmallActionIcons

```
public static void setAllowsSmallActionIcons(boolean aBoolean)
```

setBorders

```
public static void setBorders(
   int anInt,
   int anInt,
   int anInt)
```

setDisabledTextBackgroundColor

```
public static void setDisabledTextBackgroundColor(java.awt.Color aColor)
```

setEditableTextBackgroundColor

```
public static void setEditableTextBackgroundColor(java.awt.Color aColor)
```

setHighlightLabelColor

```
public static void setHighlightLabelColor(java.awt.Color aColor)
```


Draft. © Apple Computer, Inc.

setHighlightLabelFont

```
public static void setHighlightLabelFont(java.awt.Font aFont)
```

setHighlightTitleColor

```
public static void setHighlightTitleColor(java.awt.Color aColor)
```

setHighlightTitleFont

```
public static void setHighlightTitleFont(java.awt.Font aFont)
```

setLabelColor

```
public static void setLabelColor(java.awt.Color aColor)
```

setLabelFont

```
public static void setLabelFont(java.awt.Font aFont)
```

setMakeIconBackgroundsTransparent

```
public static void setMakeIconBackgroundsTransparent(boolean aBoolean)
```


Draft. © Apple Computer, Inc.

setMinimumActionButtonSize

```
public static void setMinimumActionButtonSize(java.awt.Dimension aDimension)
```

setMinimumSmallActionButtonSize

```
public static void setMinimumSmallActionButtonSize(java.awt.Dimension aDimension)
```

setMinimumSpecialActionButtonSize

```
public static void setMinimumSpecialActionButtonSize(java.awt.Dimension aDimension)
```

setOptimizesMenuAccelerators

```
public static void setOptimizesMenuAccelerators(boolean aBoolean)
```

setQueryTextBackgroundColor

```
public static void setQueryTextBackgroundColor(java.awt.Color aColor)
```

setSmallActionTitleFont

```
public static void setSmallActionTitleFont(java.awt.Font aFont)
```


Draft. © Apple Computer, Inc.

setSmallActionTitlePosition

```
public static void setSmallActionTitlePosition(int anInt)
```

setSpecialActionTitleFont

```
public static void setSpecialActionTitleFont(java.awt.Font aFont)
```

setSpecialActionTitlePosition

```
public static void setSpecialActionTitlePosition(int anInt)
```

setStandardResourceBundle

```
public static void setStandardResourceBundle(java.util.ResourceBundle aResourceBundle)
```

setTitleColor

```
public static void setTitleColor(java.awt.Color aColor)
```

setTitleFont

```
public static void setTitleFont(java.awt.Font aFont)
```


Draft. © Apple Computer, Inc.

setUsesSpecialColors

```
public static void setUsesSpecialColors(boolean aBoolean)
```

setUsesSpecialFonts

```
public static void setUsesSpecialFonts(boolean aBoolean)
```

setUsesTitleWithActionIcons

```
public static void setUsesTitleWithActionIcons(boolean aBoolean)
```

setUsesTitleWithSmallActionIcons

```
public static void setUsesTitleWithSmallActionIcons(boolean aBoolean)
```

setUsesWindowMenus

```
public static void setUsesWindowMenus(boolean aBoolean)
```

setWidgetFont

```
public static void setWidgetFont(java.awt.Font aFont)
```


Draft. © Apple Computer, Inc.

smallActionTitleFont

```
public static java.awt.Font smallActionTitleFont()
```

smallActionTitlePosition

```
public static int smallActionTitlePosition()
```

smallBorder

```
public static int smallBorder()
```

specialActionTitleFont

```
public static java.awt.Font specialActionTitleFont()
```

specialActionTitlePosition

```
public static int specialActionTitlePosition()
```

standardActionIcon

```
public static javax.swing.Icon standardActionIcon(String aString)
```


Draft. © Apple Computer, Inc.

standardResourceBundle

```
public static java.util.ResourceBundle standardResourceBundle()
```

standardSmallActionIcon

```
public static javax.swing.Icon standardSmallActionIcon(String aString)
```

titleColor

```
public static java.awt.Color titleColor()
```

titleFont

```
public static java.awt.Font titleFont()
```

usesSpecialColors

```
public static boolean usesSpecialColors()
```

usesSpecialFonts

```
public static boolean usesSpecialFonts()
```


Draft. © Apple Computer, Inc.

usesTitleWithActionIcons

```
public static boolean usesTitleWithActionIcons()
```

usesTitleWithSmallActionIcons

```
public static boolean usesTitleWithSmallActionIcons()
```

usesWindowMenus

```
public static boolean usesWindowMenus()
```

widgetFont

```
public static java.awt.Font widgetFont()
```


Draft. © Apple Computer, Inc.

EOWindowController

Inherits from:	EOComponentController : EOController : Object
Implements:	ActionListener (java.awt.event package) EOComponentController.Activation
Package:	com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

XML Tag	Default Rule System Controller Type
None (abstract class)	`windowController`

Method Types

All methods

`EOWindowController`

`generateBorderSizeForRootPaneContainerClass`

Draft. © Apple Computer, Inc.

actionPerformed

activate

activateWindow

borderSize

borderedSizeForComponentSize

componentDidBecomeInvisible

componentShouldBeResizable

componentSizeForBorderedSize

defaultActions

dispose

generateBorderSize

generateComponent

integrationComponent

minimumIntegrationComponentSize

removeTransientSubcontroller

setUsesActivationAction

setUsesActivationButton

setUsesUserDefaultsWindowLocation

setUsesUserDefaultsWindowSize

setWindowPosition

usesActivationAction

usesActivationButton

usesUserDefaultsWindowLocation

usesUserDefaultsWindowSize

windowPosition

Draft. © Apple Computer, Inc.

Constructors

EOWindowController

```
public EOWindowController(EOXMLUnarchiver anEOXMLUnarchiver)
```

Static Methods

generateBorderSizeForRootPaneContainerClass

```
protected static java.awt.Dimension generateBorderSizeForRootPaneContainerClass(
   Class aClass,
   boolean aBoolean)
```

Instance Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent anActionEvent)
```


Draft. © Apple Computer, Inc.

activate

```
public boolean activate()
```

activateWindow

```
public abstract void activateWindow()
```

borderSize

```
public java.awt.Dimension borderSize()
```

borderedSizeForComponentSize

```
public java.awt.Dimension borderedSizeForComponentSize(java.awt.Dimension aDimension)
```

componentDidBecomeInvisible

```
protected void componentDidBecomeInvisible()
```

componentShouldBeResizable

```
protected boolean componentShouldBeResizable(javax.swing.JComponent aJComponent)
```


Draft. © Apple Computer, Inc.

componentSizeForBorderedSize

```
public java.awt.Dimension componentSizeForBorderedSize(java.awt.Dimension aDimension)
```

defaultActions

```
protected NSArray defaultActions()
```

dispose

```
public void dispose()
```

generateBorderSize

```
protected java.awt.Dimension generateBorderSize()
```

generateComponent

```
protected void generateComponent()
```

integrationComponent

```
public javax.swing.JComponent integrationComponent()
```


Draft. © Apple Computer, Inc.

minimumIntegrationComponentSize

```
public java.awt.Dimension minimumIntegrationComponentSize()
```

removeTransientSubcontroller

```
protected boolean removeTransientSubcontroller(EOController anEOController)
```

setUsesActivationAction

```
public void setUsesActivationAction(boolean aBoolean)
```

setUsesActivationButton

```
public void setUsesActivationButton(boolean aBoolean)
```

setUsesUserDefaultsWindowLocation

```
public void setUsesUserDefaultsWindowLocation(boolean aBoolean)
```

setUsesUserDefaultsWindowSize

```
public void setUsesUserDefaultsWindowSize(boolean aBoolean)
```


Draft. © Apple Computer, Inc.

setWindowPosition

```
public void setWindowPosition(int anInt)
```

usesActivationAction

```
public boolean usesActivationAction()
```

usesActivationButton

```
public boolean usesActivationButton()
```

usesUserDefaultsWindowLocation

```
public boolean usesUserDefaultsWindowLocation()
```

usesUserDefaultsWindowSize

```
public boolean usesUserDefaultsWindowSize()
```

windowPosition

```
public int windowPosition()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOWindowObserver

Inherits from:	Object
Implements:	WindowListener (java.awt.event package) NSDisposable
Package:	com.apple.client.eoapplication

Class Description

Documentation for this class is forthcoming. For information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

`activateBestWindow`

`activatePreviousWindow`

`activeWindow`

`blockActiveWindowChangedNotification`

`controllerForActiveWindow`

Draft. © Apple Computer, Inc.

controllerForLatestDeactivatedWindow

controllerForWindow

dispose

latestDeactivatedWindow

previousWindowToActivate

registerWindow

registerWindow

registeredWindows

unblockActiveWindowChangedNotification

unregisterWindow

unregisterWindowOfController

visibleWindows

windowActivated

windowClosed

windowClosing

windowDeactivated

windowDeiconified

windowDidBecomeActive

windowDidBecomeInactive

windowDidBecomeInvisible

windowDidBecomeVisible

windowForController

windowIconified

windowOpened

Draft. © Apple Computer, Inc.

Instance Methods

activateBestWindow

```
public void activateBestWindow()
```

activatePreviousWindow

```
public void activatePreviousWindow()
```

activeWindow

```
public java.awt.Window activeWindow()
```

blockActiveWindowChangedNotification

```
public void blockActiveWindowChangedNotification()
```

controllerForActiveWindow

```
public EOController controllerForActiveWindow()
```


Draft. © Apple Computer, Inc.

controllerForLatestDeactivatedWindow

```
public EOController controllerForLatestDeactivatedWindow()
```

controllerForWindow

```
public EOController controllerForWindow(java.awt.Window aWindow)
```

dispose

```
public void dispose()
```

latestDeactivatedWindow

```
public java.awt.Window latestDeactivatedWindow()
```

previousWindowToActivate

```
public java.awt.Window previousWindowToActivate()
```

registerWindow

```
public void registerWindow(
   java.awt.Window aWindow,
   EOController anEOController)
```


Draft. © Apple Computer, Inc.

registerWindow

```
public void registerWindow(java.awt.Window aWindow)
```

registeredWindows

```
public NSArray registeredWindows()
```

unblockActiveWindowChangedNotification

```
public void unblockActiveWindowChangedNotification()
```

unregisterWindow

```
public void unregisterWindow(java.awt.Window aWindow)
```

unregisterWindowOfController

```
public void unregisterWindowOfController(EOController anEOController)
```

visibleWindows

```
public NSArray visibleWindows()
```


Draft. © Apple Computer, Inc.

windowActivated

```
public void windowActivated(java.awt.event.WindowEvent aWindowEvent)
```

windowClosed

```
public void windowClosed(java.awt.event.WindowEvent aWindowEvent)
```

windowClosing

```
public void windowClosing(java.awt.event.WindowEvent aWindowEvent)
```

windowDeactivated

```
public void windowDeactivated(java.awt.event.WindowEvent aWindowEvent)
```

windowDeiconified

```
public void windowDeiconified(java.awt.event.WindowEvent aWindowEvent)
```

windowDidBecomeActive

```
protected void windowDidBecomeActive(java.awt.Window aWindow)
```


Draft. © Apple Computer, Inc.

windowDidBecomeInactive

```
protected void windowDidBecomeInactive(java.awt.Window aWindow)
```

windowDidBecomeInvisible

```
protected void windowDidBecomeInvisible(java.awt.Window aWindow)
```

windowDidBecomeVisible

```
protected void windowDidBecomeVisible(java.awt.Window aWindow)
```

windowForController

```
public java.awt.Window windowForController(EOController anEOController)
```

windowIconified

```
public void windowIconified(java.awt.event.WindowEvent aWindowEvent)
```

windowOpened

```
public void windowOpened(java.awt.event.WindowEvent aWindowEvent)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOXMLUnarchiver

Inherits from: Object

Package: com.apple.client.eoapplication

Class Description

EOXMLUnarchiver objects contain the parameters used to create controllers (objects of the EOController class and its descendents) in the controller hierarchy. The parameters are determined from an XML specification sent from server.

For more information on using this class, see the book *Getting Started with Direct to Java Client*.

Method Types

Decoding objects

`decodeAlignmentForKey`

`decodeArrayForKey`

`decodeBooleanForKey`

`decodeClassForKey`

`decodeColorForKey`

Draft. © Apple Computer, Inc.

decodeDictionaryForKey

decodeEditabilityForKey

decodeFontForKey

decodeIntForKey

decodePositionForKey

decodeStringForKey

decodeValueForKey

Other methods

EOXMLUnarchiver

decodeRootObject

decodeChildren

Constructors

EOXMLUnarchiver

`public EOXMLUnarchiver(NSDictionary` *`values`*`)`

Creates an XML archiver based on the *`values`* NSDictionary.

Static Methods

decodeRootObject

`public static Object decodeRootObject(NSDictionary` *`aNSDictionary`*`)`

Decodes the top controller in an XML description, which is represented by an NSDictionary.

Draft. © Apple Computer, Inc.

Instance Methods

decodeAlignmentForKey

```
public int decodeAlignmentForKey(
   String key,
   int defaultAlignment)
```

Returns an alignment specification (`JTextField.LEFT`, `JTextField.CENTER`, or `JTextField.RIGHT`) for the *key* XML attribute. If no value for *key* is specified, returns *defaultAlignment*.

```
public int decodeAlignmentForKey(String key)
```

Returns an alignment specification (`JTextField.LEFT`, `JTextField.CENTER`, or `JTextField.RIGHT`) for the *key* XML attribute. If no value for *key* is specified, returns `JTextField.LEFT`.

decodeArrayForKey

```
public NSArray decodeArrayForKey(
   String key,
   NSArray defaultArray)
```

Returns an NSArray for the *key* XML attribute. If no value for *key* is specified, returns *defaultArray*.

```
public NSArray decodeArrayForKey(String key)
```

Returns an NSArray for the *key* XML attribute. If no value for *key* is specified, returns `null`.

decodeBooleanForKey

```
public boolean decodeBooleanForKey(
   String key,
   boolean defaultBoolean)
```

Returns a `boolean` for the *key* XML attribute. If no value for *key* is specified, returns *defaultBoolean*.

Draft. © Apple Computer, Inc.

```
public boolean decodeBooleanForKey(String key)
```

Returns a `boolean` for *`key`* XML attribute. If no value for *`key`* is specified, returns `false`.

decodeChildren

```
public NSArray decodeChildren()
```

Returns an NSArray containing the receiver's decoded children. The children are the objects created from XML tags contained in the receiver's XML description.

decodeClassForKey

```
public Class decodeClassForKey(
   String key,
   Class defaultClass)
```

Returns a Class for the *`key`* XML attribute. If no value for *`key`* is specified, returns *`defaultClass`*.

```
public Class decodeClassForKey(String key)
```

Returns a Class for the *`key`* XML attribute. If no value for *`key`* is specified, returns `null`.

decodeColorForKey

```
public java.awt.Color decodeColorForKey(
   String key,
   java.awt.Color defaultColor)
```

Returns a color (a java.awt.Color object) for the *`key`* XML attribute. If no value for *`key`* is specified, returns *`defaultColor`*.

```
public java.awt.Color decodeColorForKey(String key)
```

Returns a color (a java.awt.Color object) for the *`key`* XML attribute. If no value for *`key`* is specified, returns `null`.

Draft. © Apple Computer, Inc.

decodeDictionaryForKey

```
public NSDictionary decodeDictionaryForKey(
   String key,
   NSDictionary defaultDictionary)
```

Returns a NSDictionary for the *key* XML attribute. If no value for *key* is specified, returns *defaultDictionary*.

```
public NSDictionary decodeDictionaryForKey(String key)
```

Returns a NSDictionary for the *key* XML attribute. If no value for *key* is specified, returns `null`.

decodeEditabilityForKey

```
public int decodeEditabilityForKey(
   String key,
   int defaultEditability)
```

Returns an editability specification (`EOEditable.IfSupercontrollerEditable`, `EOEditable.AlwaysEditable`, or `EOEditable.NeverEditable`) for the *key* XML attribute. If no value for *key* is specified, returns *defaultEditibility*.

```
public int decodeEditabilityForKey(String key)
```

Returns an editability specification (`EOEditable.IfSupercontrollerEditable`, `EOEditable.AlwaysEditable`, or `EOEditable.NeverEditable`) for the *key* XML attribute. If no value for *key* is specified, returns `EOEditable.IfSupercontollerEditable`.

decodeFontForKey

```
public java.awt.Font decodeFontForKey(
   String key,
   java.awt.Font defaultFont)
```

Returns a font specification (a java.awt.Font object) for the *key* XML attribute. If no value for *key* is specified, returns *defaultFont*.

```
public java.awt.Font decodeFontForKey(String key)
```

Returns a font specification (a java.awt.Font object) for the *key* XML attribute. If no value for *key* is specified, returns `null`.

Draft. © Apple Computer, Inc.

decodeIntForKey

```
public int decodeIntForKey(
   String key,
   int defaultInt)
```

Returns an `int` for the *`key`* XML attribute. If no value for *`key`* is specified, returns *`defaultInt`*.

```
public int decodeIntForKey(String key)
```

Returns an `int` for the *`key`* XML attribute. If no value for *`key`* is specified, returns 0.

decodePositionForKey

```
public int decodePositionForKey(
   String key,
   int defaultPosition)
```

Returns a position specification (`EOComponentController.Top`, `EOComponentController.Bottom`, `EOComponentController.Left`, `EOComponentController.Right`, `EOComponentController.TopLeft`, `EOComponentController.TopRight`, `EOComponentController.BottomLeft`, or `EOComponentController.BottomRight`) for the *`key`* XML attribute. If no value for *`key`* is specified, returns *`defaultPosition`*.

```
public int decodePositionForKey(String key)
```

Returns a position specification (`EOComponentController.Top`, `EOComponentController.Bottom`, `EOComponentController.Left`, `EOComponentController.Right`, `EOComponentController.TopLeft`, `EOComponentController.TopRight`, `EOComponentController.BottomLeft`, or `EOComponentController.BottomRight`) for the *`key`* XML attribute. If no value for *`key`* is specified, returns `EOComponentController.Center`.

decodeStringForKey

```
public String decodeStringForKey(
   String key,
   String defaultString)
```

Returns a String for the *`key`* XML attribute. If no value for *`key`* is specified, returns *`defaultString`*.

Draft. © Apple Computer, Inc.

```
public String decodeStringForKey(String key)
```

Returns a String for the *key* XML attribute. If no value for *key* is specified, returns `null`.

decodeValueForKey

```
public Object decodeValueForKey(String key)
```

Returns an Object for the *key* XML attribute. If no value for *key* is specified, returns `null`.

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOAction.ActiveWindowDependentAction

Package: com.apple.client.eoapplication

Interface Description

Documentation for this interface is forthcoming. For information on using this interface, see the book *Getting Started with Direct to Java Client*.

Instance Methods

updateInContextOfActiveWindowController

```
public abstract void
   updateInContextOfActiveWindowController(EOController anEOController)
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOAction.Enabling

Implemented by: EOController

Package: com.apple.client.eoapplication

Interface Description

The EOAction.Enabling interface defines a method, `canPerformActionNamed`, which allows you to tell if an action (an EOAction object) is enabled for the receiver.

Instance Methods

canPerformActionNamed

```
public boolean canPerformActionNamed(String actionName)
```

Returns `true` if the receiver can perform an action (an EOAction object) named *`actionName`*, `false` otherwise. An EOController's implementation of this method generally returns `false` if the receiver doesn't have an action named *`actionName`* or if the *`actionName`* action is disabled.

See Also: `isActionNamedEnabled` (EOController)

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOActionWidgetController.ActionCollector

Package: com.apple.client.eoapplication

Interface Description

Documentation for this interface is forthcoming. For information on using this interface, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

```
collectedActions
```


Draft. © Apple Computer, Inc.

Instance Methods

collectedActions

```
public abstract NSArray collectedActions()
```


Draft. © Apple Computer, Inc.

EOAssociationConnector

Implemented by: EOAssociationController, EOEntityController, EORangeValueController, EOTableController

Package: com.apple.client.eoapplication

Interface Description

EOAssociationConnector is an interface that defines an object that can assume the responsibilities for connecting and disconnecting the associations of a transient subcontroller.

Instance Methods

takeResposibilityForConnectionOfAssociation

```
public abstract void
   takeResposibilityForConnectionOfAssociation(com.apple.client.eointerface.EOAssociati
   on association)
```

Invoked when one of the receiver's subcontrollers is disposed as a transient controller. This method instructs the receiver to assume responsibility for managing the subcontroller's EOAssociation, *association*.

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOComponentController.Activation

Package: com.apple.client.eoapplication

Interface Description

Documentation for this interface is forthcoming. For information on using this interface, see the book *Getting Started with Direct to Java Client*.

Instance Methods

activate

```
public abstract boolean activate()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOComponentController.EndEditing

Package: com.apple.client.eoapplication

Interface Description

Documentation for this interface is forthcoming. For information on using this interface, see the book *Getting Started with Direct to Java Client*.

Instance Methods

endEditing

```
public abstract boolean endEditing()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOComponentController.Modal

Package: com.apple.client.eoapplication

Interface Description

Documentation for this interface is forthcoming. For information on using this interface, see the book *Getting Started with Direct to Java Client*.

Method Types

All methods

```
isModal
```


Draft. © Apple Computer, Inc.

Instance Methods

isModal

```
public abstract boolean isModal()
```


Draft. © Apple Computer, Inc.

EOComponentController.ResetUserInterface

Package: com.apple.client.eoapplication

Interface Description

Documentation for this interface is forthcoming. For information on using this interface, see the book *Getting Started with Direct to Java Client*.

Instance Methods

resetUserInterface

```
public abstract void resetUserInterface()
```


Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOController.Enumeration

Implements: java.util.Enumeration

Package: com.apple.client.eoapplication

Interface Description

EOController.Enumeration is an interface that defines an enumeration that iterates over a set of EOController objects. It adds one method to the java.util.Enumeration interface: `nextController`, which simply returns the next controller in the enumeration's set. The `nextController` method saves you from having to cast the returned object to an EOController.

Use the EOController method `controllerEnumeration` to get an EOController.Enumeration. You can create three types of enumerations:

`SubcontrollersEnumeration`
: Includes all the descendants of a controller—the controller's subcontrollers, their subcontrollers, and so on down the controller hierarchy—not including the controller itself.

`SupercontrollersEnumeration`
: Includes all the ancestors of a controller—the controller's supercontroller, its supercontroller, and so on up the controller hierarchy—not including the controller itself.

`ControllerAndSubcontrollersEnumeration`
: Includes a controller and all its descendants.

Draft. © Apple Computer, Inc.

You can further restrict the controllers included in an enumeration by specifying an interface the controllers must implement in order to be included. For more information, see the method description for `controllerEnumeration` in the EOController class specification.

Instance Methods

nextController

```
public abstract EOController nextController()
```

Returns the next controller in the enumeration. Use this method instead of `nextElement` because it saves you a cast and because it's implementation is more efficient.

See Also: `controllerEnumeration` **(**EOController**)**

Draft. © Apple Computer, Inc.

EODocument

Implemented by: EODocumentController

Implements: EOObjectDisplay

Package: com.apple.client.eoapplication

Interface Description

EODocument is an interface that defines the behavior of a controller that displays and edits enterprise objects.

Instance Methods

isDocumentForGlobalID

```
public abstract boolean isDocumentForGlobalID(
   com.apple.client.eocontrol.EOGlobalID globalID,
   String entityName)
```

Returns `true` if the receiver is a document for the enterprise object associated with *`globalID`* and *`entityName`*, `false` otherwise. Typically implementations return true if the receiver's display group is displaying the specified enteprise object.

Draft. © Apple Computer, Inc.

isEdited

```
public abstract boolean isEdited()
```

Returns `true` if the receiver has unsaved edits, `false` otherwise.

save

```
public abstract boolean save()
```

Saves the receiver's edits, returning `true` on success or `false` otherwise.

saveIfUserConfirms

```
public abstract boolean saveIfUserConfirms(
   String operationTitle,
   String message)
```

If the receiver's enterprise object has been edited, opens an alert panel that allows the user to save the edits, discard the edits, or cancel the save operation. The *operationTitle* argument is used as the title of the alert panel, and *message* is used as the message in the panel. Returns `true` if the save succeeds, `false` upon failure or if the user cancels.

setEdited

```
public abstract void setEdited(boolean flag)
```

Sets the receiver's edited status according to *flag*.

Draft. © Apple Computer, Inc.

EOEditable

Implemented by: EOAssociationController, EODocumentController, EORangeValueController

Package: com.apple.client.eoapplication

Interface Description

EOEditable is an interface that defines an API for managing the editability of a branch of the controller hierarchy. EOEditable controllers usually base the editability of their user interfaces on the editability of their supercontrollers. Thus, by default all the EOEditable subcontrollers of an editable controller are also editable. To enable or disable a portion of an application's user interface, you need only message the highest level controller associated with that user interface.

Draft. © Apple Computer, Inc.

Constants

EOEditable defines the following `int` constants to identify the editability of an EOEditable controller:

Constant	Description
`NeverEditable`	The controller is never editable.
`AlwaysEditable`	The controller is always editable.
`IfSupercontrollerEditable`	The controller is editable only if its supercontroller is editable. If none of the controller's ancestors implement EOEditable, then its the same as if the controller is `AlwaysEditable`.

Instance Methods

editability

```
public abstract int editability()
```

Returns the editability of the receiver, one of `NeverEditable`, `AlwaysEditable`, or `IfSupercontrollerEditable`. The default behavior should be to return `IfSupercontrollerEditable`.

isEditable

```
public abstract boolean isEditable()
```

Returns `true` if the receiver is editable, and `false` otherwise. The default behavior should be to return `true` if the receiver is currently editable. The receiver is editable if:

- The receiver's editability is `AlwaysEditable`.

Draft. © Apple Computer, Inc.

- The receiver's editability is `IfSupercontrollerEditable` and sending `isEditable` to the first EOEditable ancestor of the receiver returns `true`.

setEditability

```
public abstract void setEditability(int editability)
```

Sets the receiver's editability to *editability*, one of `NeverEditable`, `AlwaysEditable`, or `IfSupercontrollerEditable`.

supercontrollerEditabilityDidChange

```
public abstract void supercontrollerEditabilityDidChange()
```

Invoked to notify the receiver that the editability of its supercontroller changed, giving the receiver the opportunity to update its user interface to match the editability of the supercontroller.

takeResposibilityForEditabilityOfAssociation

```
public abstract void
   takeResposibilityForEditabilityOfAssociation(com.apple.client.eointerface.EOAssociat
   ion association)
```

Invoked when one of the receiver's subcontrollers is disposed as a transient controller. This method instructs the receiver to assume responsibility for managing the editability of the subcontroller's EOAssociation, *association*.

Draft. © Apple Computer, Inc.

Draft. © Apple Computer, Inc.

EOModalDialogController.ModalActions

Package: com.apple.client.eoapplication

Interface Description

Documentation for this interface is forthcoming. For information on using this interface, see the book *Getting Started with Direct to Java Client*.

Instance Methods

cancel

```
public abstract void cancel()
```

modalDialogShouldClose

```
public abstract boolean modalDialogShouldClose()
```


Draft. © Apple Computer, Inc.

ok

```
public abstract boolean ok()
```


Draft. © Apple Computer, Inc.

EOObjectDisplay

Implemented by: EOEntityController

Package: com.apple.client.eoapplication

Interface Description

EOObjectDisplay is an interface that defines the behavior of a controller that displays enterprise objects using an EODisplayGroup.

Instance Methods

controllerDisplayGroup

```
public abstract com.apple.client.eointerface.EODisplayGroup controllerDisplayGroup()
```

Returns a display group containing the receiver—an EOController or subclass. This display group can be used to connect controller methods to the user interface.

Draft. © Apple Computer, Inc.

displayGroup

```
public abstract com.apple.client.eointerface.EODisplayGroup displayGroup()
```

Returns the display group the receiver uses to display and edit the properties of its enterprise objects.

editingContext

```
public abstract com.apple.client.eocontrol.EOEditingContext editingContext()
```

Returns the editing context the receiver uses to manage the graph of its enterprise objects.

entityName

```
public abstract String entityName()
```

Returns the name of the entity that describes the enterprise objects the receiver displays with its display group.

Draft. © Apple Computer, Inc.

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software.

Line art was created using Adobe™ Illustrator and Adobe Photoshop.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Adobe Letter Gothic.